AD-A072 917 DATA/WARE DEVELOPMENT INC SAN DIEGO CALIF F/G 9/2

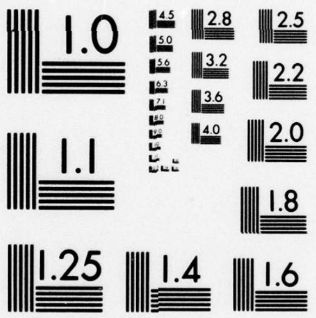
ADVANCED DIGITAL TV SYSTEM.(U)

FEB 79 P J ERDELSKY, R V KEELE, G G MURRAY F33615-77-C-1198

UNCLASSIFIED AFAL-TR-79-1006 NL

1 OF 4

AD A072917


1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

10

AFAL-TR-79-1006

ADVANCED DIGITAL TV SYSTEM

AUTHOR-(s)
Philip J. Erdelsky, Richard V. Keele,
G. Graham Murray, Ronald B. Weiss

Data/Ware Development, Inc.
4204 Sorrento Valley Boulevard
San Diego, California 92121

February, 1979

TECHNICAL REPORT AFAL-TR-79-1006

Final Report for period June 1977 - September 1977

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

Ronald A. Belt

RONALD A. BELT, Acting Chief
Processor Technology Group

FOR THE COMMANDER

Stanley E. Wagner

STANLEY E. WAGNER, Chief
Microelectronics Branch
Electronic Technology Division

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 18 AFAL-TR-79-1006 14	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 ADVANCED DIGITAL TV SYSTEM.		5. TYPE OF REPORT & PERIOD COVERED 9 Final Rep[redacted]t. June-Sept[redacted]77
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Philip J. Erdelsky, Richard V. Keele, G. Graham Murray, Ronald B. Weiss		8. CONTRACT OR GRANT NUMBER(s) 15 F33615-77-C-1198
9. PERFORMING ORGANIZATION NAME AND ADDRESS Data/Ware Development, Inc. 4204 Sorrento Valley Boulevard San Diego, CA 92121		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 16 6096 05 86 17 Ø5
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory AFAL/DHE-1 WPAFB, OH 45433		12. REPORT DATE 11 Febr[redacted]979
		13. NUMBER OF PAGES 341
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 350p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

62204F

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Programmable Digital Processors	Image Processing
Signal Processing	Digital Processing
Cosine Transform	Pipelined Processors
Microprocessors	Video Signal Processing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A laboratory version of a programmable digital processor system for TV bandwidth reduction was designed and placed in operation. The system consists of a TV camera-A/D converter unit, two pipelined microprocessor units, and a frame store-D/A-TV display unit interfaced to a PDP-11/04 minicomputer controller in such a manner that data transfers can be effected between any two units over the PDP-11/04 Unibus under software control of the CPU. The pipelined

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

392 006 Gm

microprocessor units are software programmable via local program stores which are loaded over the PDP-11/04 Unibus. Each pipe-lined processor is twelve bits wide and uses three AMD 2901 bit-slice microprocessor chips in conjunction with a 12x12 bit hard-ware multiplier for high speed computation. Each processor has a program memory capable of storing 1024 program instructions and a data memory of 16 registers. Pipelining of the fetch, store ALU, and multiplication operations allow a thruput rate of ap-proximately 24 MOPS, which is sufficient for performing the DCT/DPCM bandwidth reduction algorithm at 3.75 frames per second on a 256 pixel image. By using eight vertical stripes instead of full frame video, a frame store memory at the A/D input is not required. The final report contains a description of the hardware design, software control packages, software development packages, and source code for executing the DCT/DPCM bandwidth reduction/expansion algorithms.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont'd)

LIST OF TABLES

SECTION I

INTRODUCTION

The Model D/W 1240 Digital Video Processor System has been designed to permit the realistic simulation of remote capture of TV pictures, their digitizing, compression, transmission to a remote site, their processing, conversion back to analog form, and display on a monitor. One immediate application is in Remotely Piloted Vehicles, which are unmanned and carry a small TV camera plus communications equipment for transmission to a ground station. See Fig. 1 which depicts the equipment carried in the RPV and that required at the Receiving Site.

An important requirement is for data compression. The technique which many investigators consider optimum in this application is the use of the Discrete Cosine Transform in the horizontal direction together with Differential Pulse Code Modulation line to line. Greatly reduced data rates can be achieved which permit satisfactory reconstructed images on the ground when the inverse transformation is carried out.

TV images are treated as 256 lines of 256 pixels (picture elements), the TV camera video being converted to 6 bits of accuracy. Frame slow down by a factor of 8 combined with the Discrete Cosine Transform (DCT) along a TV line and Differential Pulse Code Modulation (DPCM) line to line can reduce the data rate to as low as 200 kbps. At the receiving site the inverse transformations are performed to recover the im age, which is stabilized in a digital Frame Store Memory (FSM) during conversion back to analog for display on the TV monitor.

In the system of Fig. 2, the PDP-11 UNIBUS and not the modems depicted in Fig. 1, acts as the interconnecting communications channel. In order to facilitate laboratory studies in which the

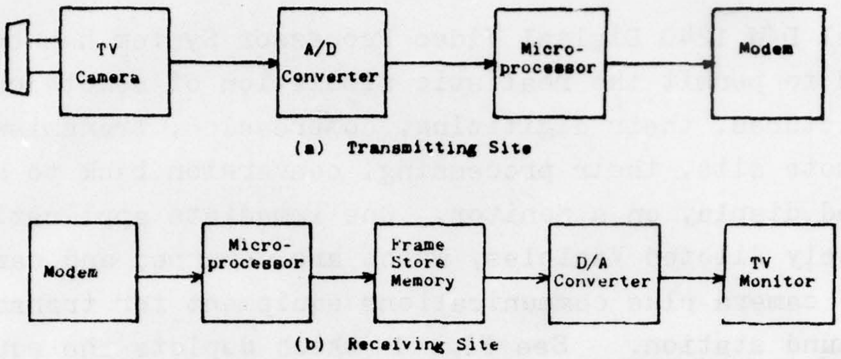


Figure 1. Operational System Block Diagram

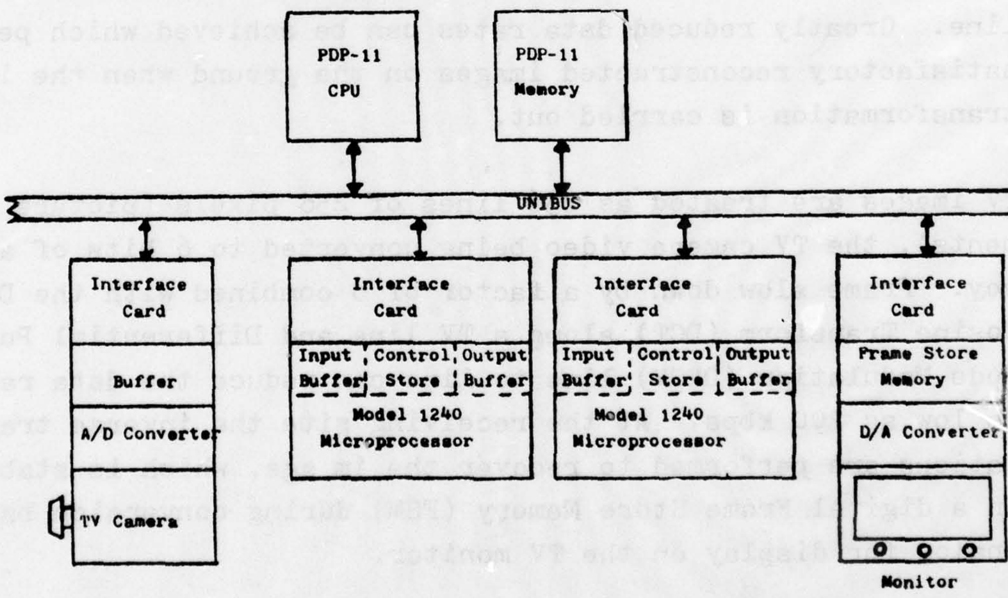


Figure 2. Laboratory System Block Diagram

minicomputer is able to probe each subsystem, the following units are accessible via the UNIBUS: (1) the TV camera and A/D converter combination, (2) the microprocessor at the transmitting site, (3) the microprocessor at the receiving site, and (4) the Frame Store Memory. In addition, the 65,000 8-bit bytes of the FSM, organized as 32,000 16-bit words, have been memory mapped so that they appear to the PDP-11 as 4096 word pages of its own memory. Thus it is very easy for the computer to investigate the effects of various algorithms upon the quality of the image. Note that the page size corresponds to the stripe dimensions of 32 pixels in width and 256 lines. Just as the minicomputer has access to the FSM, so the microprocessors can read and write the core memory of the PDP-11. As set up under software control, each microprocessor can fetch pixels (two to a word) from either the TV camera or the FSM or it can fetch transform coefficients from core memory. Also, the microprocessor can store 16 pixel pairs in FSM or send 32 12-bit coefficients (transform results) to the other microprocessor or to core memory. Core memory thus is available to the microprocessors for storing a stripe's worth of coefficients. By means of a special addressing feature, the microprocessor can address these coefficients in a "corner turned" manner to permit 2-dimensional transforms (e.g. DCT/DCT, Fourier/ Fourier, etc.). Because the coefficients are available in core memory, the PDP-11 is also able to examine or to manipulate them.

Activity along the UNIBUS is controlled by set-up operations of the PDP-11. In one mode, for example, the TV camera is commanded to send a stripe 32 pixels wide -- the stripe being selectable -- from each field to the first microprocessor. The latter will carry out the forward transforms on each line and transfer the "coefficients" which result from the DCT/DPCM operation to the second microprocessor which performs the inverse processing. The recovered pixels are next passed to the FSM to update the TV display. This system runs real time within the limitation of processing only one stripe per field time.

1.1 System Description

Rather than assembling the rather limited configuration of Figure 1, a much more powerful, general-purpose system was placed in operation. As shown in Figure 2, it replaces the modems with the UNIBUS of the PDP-11 minicomputer -- at the same time permitting the minicomputer to have control over all the subsystems and the mode of operation.

The two key elements are the microprocessors which are 12-bit units assembled from the Schottky TTL Am 2901 4-bit-slice microprocessor chips. Capable of performing any arithmetic or logical operation (including multiply) in 150 ns, each Model 1240 microprocessor runs under microprogram control from a 48-bit wide RAM microstore to which the PDP-11 has direct access.

Because the system is entirely microcoded, it can serve to study any of a wide variety of compression algorithms. The initial one demonstrated is that due to Habibi, sometimes called a hybrid technique because it is a combination DCT/DPCM. It has been shown by simulation to give performance close to the optimum of the Karhunen-Loeve transform.

It was convenient to place all the system elements on the UNIBUS to permit flexibility of operation. In the PDP-11 any two devices attached to the UNIBUS (including the CPU and high-speed memory) can communicate, provided that one acts as Bus Master and the other as Bus Slave. The CPU is always a Master and memory a Slave. The Model 1240 system makes the TV camera a Slave, but the Frame Store Memory and the microprocessors can be either Masters or Slaves.

The UNIBUS permits a maximum rate of transfer of 2.5 megawords (5.0 megabytes) per sec. This is sufficient to allow sending the 256 pixels of a TV line from the camera to any destination in the available 63.5 microseconds. In particular, an entire field of 256x256 pixels can be transferred to the 65,536 byte FSM.

The mode which is most important is that in which a "stripe" 32 pixels wide and 256 lines deep is transmitted during a field time. This is an 8 to 1 field slow down which gives acceptable results at the ground station in typical airborne applications. The stripes are processed from left to right, each stripe being updated at a 7.5 times a second rate.

When the system is in the stripe mode, 32 pixels from each line are sent to the first microprocessor which performs the DCT/DPCM computation on the entire stripe. Note that a 32-point DCT computation, followed by the DPCM, must be executed by the microprocessor in 63.5 microseconds or less. Employing an algorithm for the DCT superior to any published which is due Data/Ware, the DCT can be computed with 194 additions and 115 multiplications. With its pipelined architecture and fast 150 ns microinstruction time, the Model 1240 microprocessor is able to perform these computations in the allocated time.

1.2 DCT/DPCM Computations

A new very efficient formulation of the DCT algorithm is employed in the system. If $g_0, g_1, \ldots, g_{31}$ are the numbers representing the pixel values in one "stripe" of a TV line, the Discrete Cosine Transform (DCT) is given by

$$G_k = 2\sum_{j=0}^{31} g_j \cos\left[(j+\tfrac{1}{2})k\theta\right] \qquad (1)$$

$$k=0, 1, \ldots, 31$$

where $\theta = 2\pi/64$. By combining complex conjugate terms,

$$G_k = w^{-\frac{1}{2}k}\sum_{j=0}^{63} a_j \; w^{-jk} \qquad (2)$$

where $w = e^{i\theta}$, i is the square root of -1, and

$$a_j = \begin{cases} g_j & \text{if } j \leq 31 \\ g_{63-j} & \text{if } j \geq 32 \end{cases}$$

This formulation can be slightly restated:

Define

$$A_k = \sum_{j=0}^{63} a_j \quad w^{-jk} \tag{3}$$

Then

$$G_k = w^{-\frac{1}{2}k} \quad A_k \tag{4}$$

where this last step is referred to as a rotation. As a result of an algorithm due to Data/Ware Development, the above procedure can be simplified to the computation of complex quantities, B_k, which must be somewhat similarly rotated in order to yield the desired G_k.

The main computation to be performed is that of the B_k, the flow chart for which is shown in Figure 3. Four types of butterflies or semi-butterflies can be distinguished, as indicated in Table 1. Of interest is the total number of additions and multiplications required to compute the B_k quantities -- 164 and 54, respectively.

After the final step of rotating the B_k and forming the G_k, the DCT computation is complete. Next the DPCM is performed in the microprocessor upon the coefficients of each line segment. The quantization of the differences between a coefficient and its predicted value will depend upon the particular coefficient because higher frequency coefficients have smaller variances. The usual criterion for establishing quantization levels is to make all quantization levels equally likely. This can be done under the assumption that the amplitudes of the coefficient differences are exponentially distributed.

Figure 4 is a schematic representation of the DPCM computation. $G_k^{(n)}$ is the kth coefficient for the nth line as it is received from the DCT algorithm. The corresponding coefficient from the preceding line is attenuated by the quantity α, which lies between 0 and 1, and subtracted from the new coefficient. The

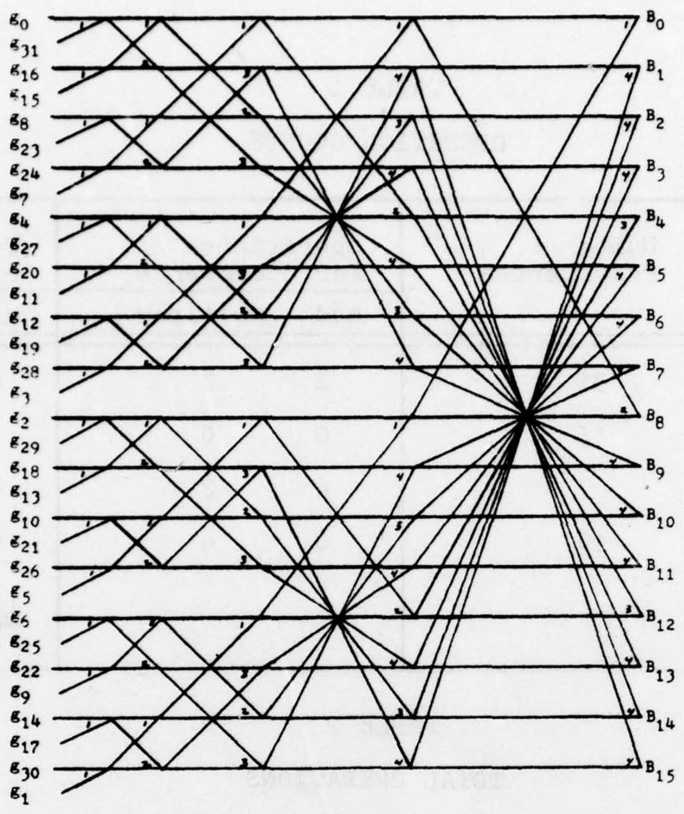


Figure 3. New DCT Algorithm

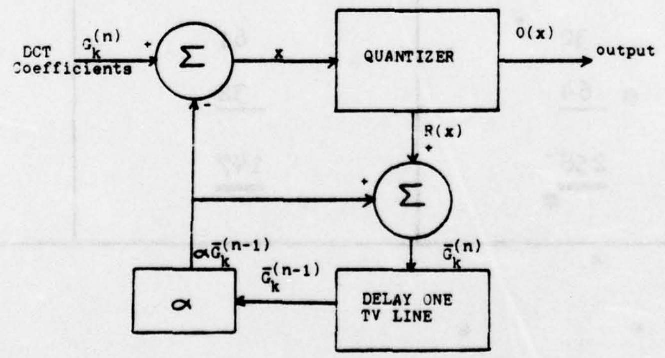


Figure 4. DPCM Computation

TABLE 1

OPERATION COUNTS

Butterfly Type	Number of Occurrences	Operations Per Occurrence		Total Number of Operations	
		add	multiply	add	multiply
1	31	2	0	62	0
2	15	0	0	0	0
3	7	6	2	42	14
4	10	6	4	60	40
Totals				164	54

TABLE 2

TOTAL OPERATIONS

	Additions	Multiplications	Total
B_k	164	54	218
Rotations	30	61	91
DPCM	64	32	96
Totals	258	147	405

difference, x, is the input to the quantizer, whose output depends upon x and upon k (the frequency of the coefficient). O(x) is the code word sent to the receiving site, and R(x) is the rounded value of the difference. The smoothed estimate of the kth coefficient is given by

$$\overline{G}_k^{(n)} = R(x) + \alpha\overline{G}_k^{(n-1)} \qquad (5)$$

$$\text{where} \quad x = G_k^{(n)} - \alpha\overline{G}_k^{(n-1)} \qquad (6)$$

The starting value of $\overline{G}_k^{(n-1)}$ is not important because α is less than 1 and the initial value is attenuated to zero.

1.3 System Implementation

Each major subsystem of Figure 1-2 is interfaced to the UNIBUS via a plug-in card so that the various key registers of the subsystem appear as high-speed (core) memory locations to the PDP-11 CPU. This means that the PDP-11 can directly modify the control store contents within each microprocessor and interrogate or modify various registers. Thus the microprocessor is controlled not by a front panel but rather by an operating system within the PDP-11 computer. Additional registers within the interface card itself establish the overall mode of operation. Each microprocessor can be set up to act as Bus Master or Bus Slave. Typically the first microprocessor will act as Master in fetching pixels from the TV camera and in sending DCT/DPCM "coefficients" to the second microprocessor.

This requires the second microprocessor to be commanded to accept coefficients as a Slave but to send the recovered pixels to the FSM as a Master.

Another possibility is for the first microprocessor to send the coefficients to the core memory of the PDP-11, from which the second microprocessor can fetch them. This is convenient for 2-D work in which the first microprocessor performs the horizontal DCT, Fourier, Hadamard, or Haar transform, and the second microprocessor performs the vertical transform. The coefficients can now be manipulated by the PDP-11 either for enhancement or for redundancy reduction. This would be followed by the inverse transforms. Implied in the above is the ability of the microprocessor interface card to carry out "corner-turned" addressing of core memory when required for 2-D work.

Because the PDP-11 has access to the interface cards, it controls completely the mode of operation. The end of each field time is signalled to the PDP-11 as an interrupt. During vertical retrace time the CPU can set up the desired processing for the next field. For example, the CPU must change the stripe by incrementing various registers in the interface cards.

The FSM is regarded by the PDP-11 as a 32,768 word memory to which it has access as 8 pages, each of 4,096 words. This makes it quite convenient for the CPU to interrogate and manipulate pixels within the FSM. Conversely, it was mentioned above that the microprocessors can access the PDP-11 core memory, which greatly reduces the need for transferring blocks of data within the system.

The system can capture an entire TV field rather than operating in the stripe mode. This would avoid the problem of edge effects from assembling 8 stripes into a single picture. Also, provision has been made for adding a second FSM. This would simulate a mode of operation in which the transmitting site seizes a field before compressing it.

In an actual application, the transmitting site would have to send a synchronizing signal. For laboratory work the synchronization is provided by a Fairchild 3262A sync generator chip whose outputs are fed to the camera and monitor. Through the use of FIFO memories for both input and output, the microprocessors can run independently from the synchronizing signals. When the input FIFO is empty or the output FIFO full, the microprocessor suspends operation.

Although it clearly results in a processing slow-down, it is possible to operate the system with a single microprocessor which will both compress the data and expand it. It is necessary to bring the field directly from the camera to the FSM. This data is then read out to the microprocessor for compression, followed by expansion. The processed pixels are then stored back in the FSM for viewing. When done dynamically with a single FSM, the TV monitor alternately shows the original pixels and the processed pixels unless the system has two FSM's.

1.4 Model 1240 Microprocessor

Assembled from three Am 2901 4-bit slice Schottky TTL microprocessor chips, the Model 1240 is a very high-performance processor ideally suited for signal processing applications. The Am 2901 ALU accepts two inputs, which typically come from the 16 general registers of an internal dual-port RAM. Each microinstruction combines arithmetically or logically two operands and places the result, unshifted or shifted right or left, back in the second general register. This result can also be put on the data output bus. Instead of taking both operands from general registers, the Am 2901 can alternatively accept one input from the Direct Data bus. System cycle times as short as 150 ns have been demonstrated.

In the Model 1240, pipeline architecture implemented with Schottky TTL circuit provides very fast processing. The Am2901 elements and other essential subsystems, including scratch pad memory and asynchronous multiplier, have been carefully organized to produce a system structure that can be programmed using a high degree of pipelining. Propagation delays through the various system elements are matched using data latches where necessary and numerous data paths are available so that parallel transfers are possible.

To realize the full potential of this pipeling architecture programs are typically coded entirely in microcode using straight line programming. Using this method a maximum number of parallel operations can occur and the inefficiencies attributable to a hierarchy of code, and resultant program branches, are eliminated.

The Am 2901 combined with a ROM microcontrol unit provides a flexible system whose performance cannot be matched by an MSI implementation. The MSI equivalent of the Am 2901 chip alone would require 15-20 16-pin devices at a typical operating power of 3.6 watts. The Am 2901 40-pin device has a typical power dissipation of only 0.97 watts.

The basic elements comprising the microprocessor are depicted in Figure 5. At the heart of the system is the microprocessor that performs the sum and difference calculations of the DCT and the DPCM algorithms, stores temporary results in a 16 word dual port memory, and transfers data to different elements of the system. An additional 64 words of memory are provided by 64x9 RAMs to allow adequate storage of certain DCT coefficients and 32 DPCM previous values. A 12x12 multiplier supplies the high speed multiply capability that is imposed by real time processing and a quantizer/dequantizer processes rounded values for the DPCM algorithm. System I/O is accomplished through an input interface and an output interface that buffer data and provide synchronization and control. All algorithms reside in the memory of the ROM-based control unit that manages the operation of each system element.

The Am 2901's operate in parallel to give a 12-bit word length. A "look ahead carry" generator is included to speed up arithmetic operations. Additional data storage is provided by the RAM memory. Direct input data to the Am 2901's is routed via an 8 to 1 multiplexer to a 12-bit input register. The input register is located at the direct data inputs to assure that processing through the longest path (the Am 2901) can be completed within a cycle time. The typical cycle time for this path is approximately 120 ns. Output data from the Am 2901 goes directly to the other systems elements.

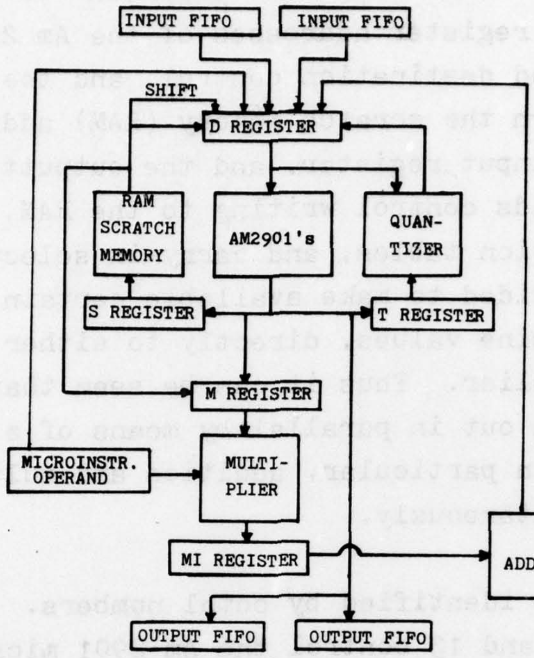


Figure 5. Model 1240 Microprocessor

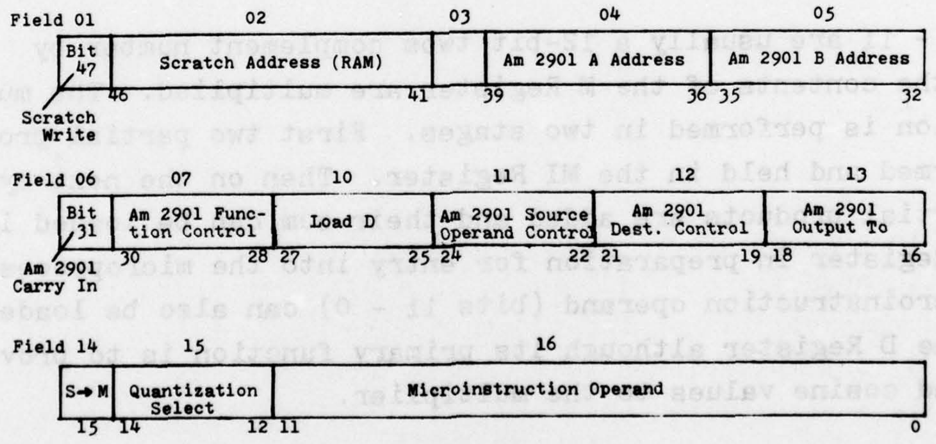


Figure 6. Microinstruction Format

1.5 Microinstruction Format

The microinstruction, shown in Figure 6, is 48 bits in length and consists of 14 fields. The more important of these control the A and B general register addresses of the Am 2901 (4 bits each), the source and destination control, and the function. Certain fields govern the scratch memory (RAM) address, the loading into the D input register, and the outputting from the Am 2901. Other fields control writing to the RAM, the selection of various quantization tables, and carry-in selection. A 12-bit field is provided to make available certain operands, such as sine and cosine values, directly to either the D register or to the multiplier. Thus it can be seen that many operations can be carried out in parallel by means of a single microinstruction. In particular, addition and multiplication can take place simultaneously.

Note that fields are identified by octal numbers. Fields 04, 05, 06, 07, 11, 12, and 13 control the Am 2901 microprocessor system. Field 02 (bits 46 - 41) provides the Scratch Memory (RAM) address. Bit 47 is 1 whenever the Scratch Memory is being written. If bit 15 is 1, the Scratch Memory output is loaded into the M Register in preparation for a multiplication.

Bits 0 - 11 are usually a 12-bit twos complement number by which the contents of the M Register are multiplied. The multiplication is performed in two stages. First two partial products are formed and held in the MI Register. Then on the next cycle, the partial products are added and their sum can be loaded into the D Register in preparation for entry into the microprocessor. The microinstruction operand (bits 11 - 0) can also be loaded into the D Register although its primary function is to provide sine and cosine values to the multiplier.

Bits 14 - 12 are used to form part of the Quantizer Table address in the DPCM process. The contents of the T Register supply the main portion of the Quantizer address, T being loaded from the microprocessor with the difference quantity, x, of Figure 4. In a sense bits 14 - 12 select a particular table, as dictated by the order of the coefficient being currently processed. Register D can be loaded from any of seven sources, as controlled by bits 27 - 15.

The Am 2901 microprocessor output can be directed to any of five destinations under the control of bits 18 - 16. There are, however, two bit patterns which are also used for other purposes. One causes the contents of the D Register to be written into the Quantizer Table for initial loading in a laboratory environment (where this table is stored in a RAM). The other causes transfer of microcontrol. Unless this latter bit pattern is present, microcontrol passes to the next microinstruction.

1.6 Microcoding the DCT/DPCM

It is now possible to define the microprogramming task for the Model 1240 in its entirety. Table 2 summarizes the additions and multiplications required to carry out the B_k calculation, the rotations, and the DPCM. The grand total is 405 arithmetic operations, of which 147 are multiplications. However, the distinction between addition and multiplication is not so important because the Model 1240 is able to accomplish either in 150 nsec. In order to carry out 405 microinstructions in the alloted line time of 63.5 usec, the Model 1240 must be able to execute an operation in 157 nsec. Furthermore, the coding must be extremely efficient. The latter is in fact true because the microinstructions control many paths at once.

In the design of the 1240 the most important consideration was this ability to overlap operations while at the same time not requiring an excessive number of components. The final arrangement of the microporgram is one in which the real parts of the computation are stored in the general registers and the imaginary parts in an external RAM. All of the sixteen Am 2901 general registers are needed in the computation, and the availability of the Q Register is a definite asset.

The entire code to carry out the 405 arithmetic operations is less than 405 microinstructions, which is an indication of the efficiency of design. The program is written entirely in-line. There are no tests and no jump instructions. Input data is brought in via a FIFO, and output is handled in the same manner. Hardware tests that there is data available in the input FIFO and that the output FIFO is not full. If these conditions are not met, the microprocessor will wait, which is the basic method of synchronization.

Special code is required for each of the butterflies of Table 1. The most difficult case is that in which 6 additions and 4 multiplies must be performed. The coding for this will be presented in order to demonstrate the basic concept of microcoding.

The complex number (d,e) is to be rotated through an angle θ and added and subtracted with the complex number (b,c). b and d will be in General Register (GR) while c and e will be stored in RAM. The computation to be performed is

$$b \pm (d\cos\theta - e\sin\theta) \rightarrow GR,$$

$$\pm c + (d\sin\theta + e\cos\theta) \rightarrow RAM.$$

sinθ and cosθ are stored directly within the microinstruction and are automatically supplied to the multiplier. The following notation (see Figure 4) is used: M is the input (multiplicand) to the multiplier and MIR is the output register. D is the input register to the microprocessor and S is the input register to the scratch memory (RAM). The microcode is as follows:

Microinstruction	Interpretation
RAM → M	e → M
GR → M, M → MIR	d → M
MIR → D, M → MIR	$e\sin\theta$ → D
D → Q, MIR → D, M → Mir	$e\sin\theta$ → Q, $d\cos\theta$ → D
D - Q → Q, MIR → D	$d\cos\theta - e\sin\theta$ → Q, $d\sin\theta$ → D
GR - Q → GR, RAM → M	b - Q → GR, e → M
GR + Q → GR, M → MIR	b + Q → GR
D → Q, MIR → D	$d\sin\theta$ → Q, $e\cos\theta$ → D
Q + D → Q, RAM → D	Q + $e\cos\theta$ → Q, c → D
D + Q → S (RAM → M)	c + Q → S (e → M)
-D + Q → S, S → RAM	-c + Q → S, c + Q → RAM
S → RAM (GR → M, M → MIR)	-c + Q → RAM, (d → M)

There are 12 microinstructions but two (parentheses) are look-ahead and so overlapped with the next butterfly. It follows that the 6 additions and 4 multiplies have been accomplished in 10 microinstructions, which was the objective. The logical organization of Figure 5 has been carefully arranged to optimize the flow of data and intermediary results. This is made possible by the microprogrammed organization which can control several operations at once.

1.7 System Modes

Possible system modes are shown in Figure 7. The two important modes are those numbered 1 and 2 in this figure. The first corresponds to a single microprocessor system and the second to two microprocessors, which is the delivered configuration.

The burden in order to be able to implement these modes lies on the microprocessor BIC. It is true that the PDP-11 must specify the mode it wishes and must set up the Control and Status Register in the microprocessor BIC, but the switching and desicion making is the responsibility of the latter.

System Mode	Microprocessor Input	Input Control	Microprocessor Output	Output Control
1. Configuration 1 (Single processor)	16 pp (pixel pairs) from FSM	at horiz. sync but every other line	16 pp to the FSM	during horiz. or vertical retrace
2. Configuration 2				
2a. 1st micro-processor	16 pp from the TV camera BIC or from FSM	Following Gate Enable, at horiz sync, every line	32 12-bit coef-ficients to 2nd processor	after input, not horiz re-trace, or dur-ing vert retrac
2b. 2nd micro-processor	32 coefficients from 1st micro-processor	1st microproces-sor as Master will fill FIFO	16 pp to the FSM	during horiz. or vertical retrace

Figure 7. System Modes

The required flexibility of the microprocessor BIC (MP BIC) can be seen from Figure 7. The column under microprocessor input indicates that the MP BIC must be able to act as a bus Master and take either 16 words or 32 words from the FSM, TV BIC, or the core memory. However, 16 words are always taken from the FSM and TV BIC while 32 words are always taken from core memory. Simularly, as Master the MP BIC must be able to send 16 words to the FSM or 32 words to the core memory or 32 words to the second microprocessor. As a Slave device (when it is the second microprocessor) it must be able to accept 32 words. This information is summarized in Figure 1-8. As a result the MP BIC is rather sophisticated.

In order to decouple the microprocessor from the UNIBUS, the MP BIC communicates with the microprocessor Input FIFO and Output FIFO. These are 64 word memories, which can operate both in the 8-bit byte mode and in a full 16- or 12-bit mode. There is no distinction between the full 16-bit and 12-bit modes because the UNIBUS is 16 bits in width and always supplies a word of this size, as used in this application, while the microprocessor is 12 bits wide and hence rejects 4 of the 16 bits.

Acting as Bus Master

Input:

16 words from camera BIC

16 words from FSM BIC

32 words from core memory

Output:

16 words to FSM

32 words to 2nd microprocessor

32 words to core memory

Acting as Bus Slave

Input:

32 words from 1st microprocessor

Figure 8. Microprocessor BIC I/O

SECTION II

SYNCHRONIZATION AND VIDEO SUBSYSTEMS

The System Block Diagrams of Fig. 2 and 3 show a Sync Subsystem as well as TV camera and Analog-to-Digital Converter on the input and a Digital-to-Analog Converter and TV Monitor at the output. Although the heart of the Model D/W 1240 Digital Video Processor System consists of digital circuitry, it is nevertheless critically dependent upon the analog portions. This section is concerned with brief descriptions of the following subsystems:

1) **Sync Generator**
2) **ADC Buffer**
3) **Video Generator.**

The TV camera included as part of the system is the Cohu 4410. Its features include automatic sensitivity and black level control, resolution of 800 lines in the center, ability to resolve all 10 shades of gray on EIA TV Test Chart with only 0.5 footcandle illumination using the Type 8541 Vidicon. It provides 1.0 V peak-to-peak composite signals in accordance with EIA RS-170 specification. Data/Ware supplies this camera with the -400 genlock option so that the camera can be locked in phase with an external sync.

The monitor supplied is the Conrac SNA series in the 17-inch CRT size. It is all solid-state with regulated low voltage and stabilized high voltage supplies. Scan rates are the conventional 525/60 U.S.A. ones, and it provides 800 line center resolution minimum with good gray scale rendition. The horizontal and vertical sizes are adjustable. The monitor can be operated with video tape recorders. It will accept either composite video or separate video and mixed sync. A DC restorer switch allows selection of 100% or zero DC restoration.

Both commercial and militarized ADC's are available which can meet the requirement for conversion of analog to digital at a 5 MHz sample rate. Data/Ware employs an ADC which meets speci-

fic customer requirements. Considerable flexibility is possible with respect to bits of precision, method of control, number representation, voltage levels, etc. The particular ADC employed at present provides ECL voltage levels at the output. The video input must be conditioned suitably from the camera.

2.1 Sync Generator

This single card provides the basic TV timing signals necessary in order to synchronize the analog and digital portions of the system. The block diagram of Fig. 9 shows that the frequency reference is a 19.656 MHz crystal oscillator. When divided by 4, this produces the pixel clock (PCLK), which is a 4.914 MHz signal. After a further division by 6, the resulting 819 KHz signal is provided as input to a phase-locked loop. The latter provides a multiplication by 5, which results in a 4.095 MHz signal next divided by 2 as input to the Fairchild 3262A TV sync generator chip.

This Fairchild IC is extremely useful for providing sychronizing signals in TV systems. Among its outputs are

1) Vertical Drive (VD)
2) Odd field pulse
3) Even field pulse
4) Composite blanking
5) Composite sync

When these signals plus PCLK are made available to the various subsystems, the total system can be synchronized properly.

2.2 ADC Buffer

The purpose of this card is to adjust certain voltages and signal levels of the ADC to be compatible with other subsystems. The video buffer section accepts 75 ohm video from the camera. It must clip the sync, set the signal level, and condition the signal amplitude. It also provides a 50 ohm output for the

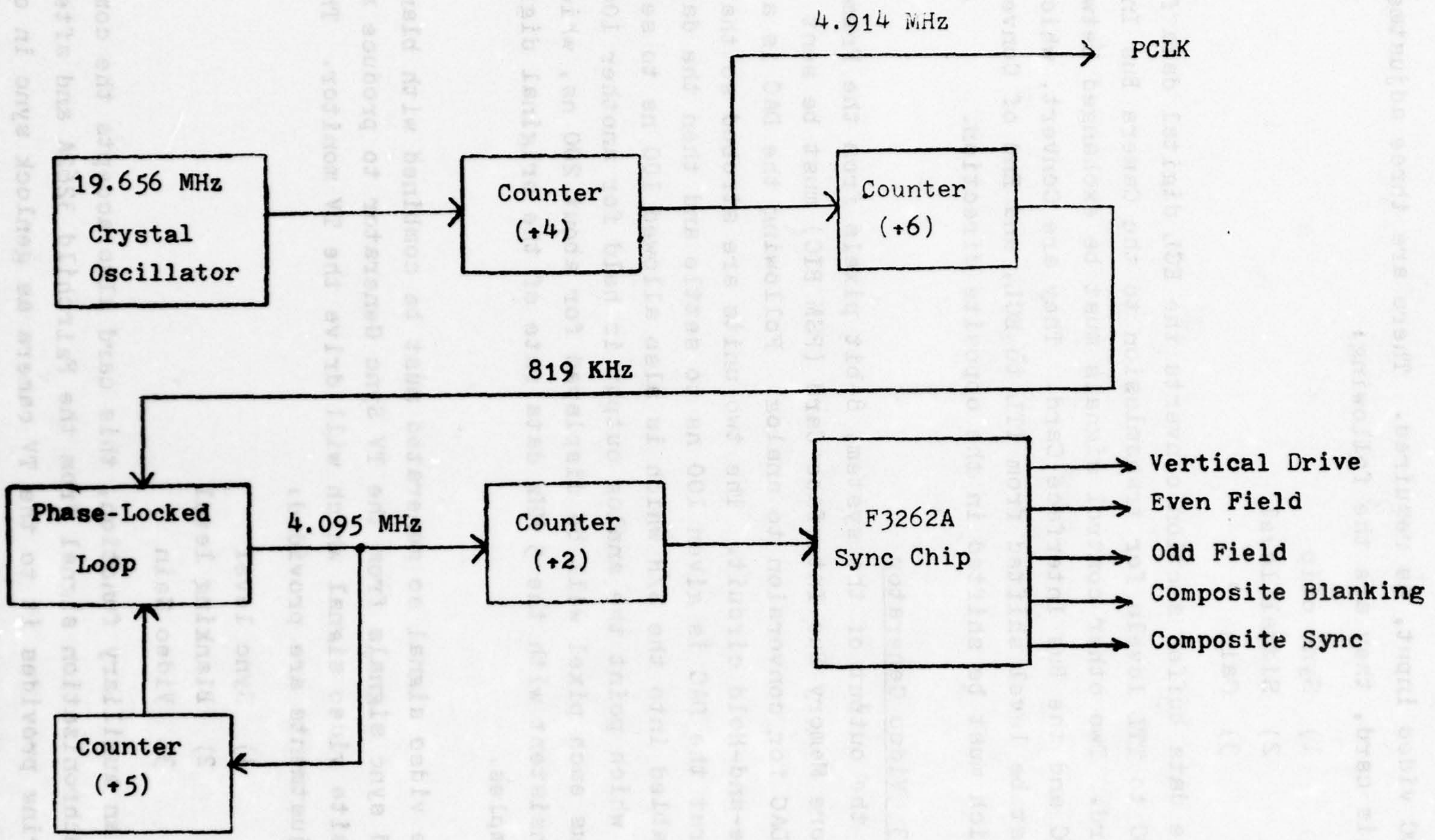


Figure 9 Sync System Block Diagram

ADC video input, as required. There are three adjustments on this card, they are the following:

1) Sync clip
2) Signal level
3) Gain

The data buffer section converts the ECL digital data from the ADC to TTL levels for transmission to the Camera Bus Interface Card. Two other control signals must be exchanged between the ADC and the Bus Interface Card. They are Convert, which must be level shifted from TTL to ECL, and End of Conversion which must be shifted in the opposite direction.

2.3 Video Generator

At the output of the system, 8-bit pixels from the Frame Store Memory Bus Interface Card (FSM BIC) must be sent to a DAC for conversion to analog. Following the DAC is a Sample-and-Hold circuit. The two units are strobed so that first the DAC is given 100 ns to settle and then the data is enabled into the S/H which is also allowed 100 ns to settle, at which point the analog output is held for another 100 ns. Thus each pixel will be displayed for about 200 ns, which is consistent with the 5 MHz data rate of the original digitized samples.

The video signal so generated must be combined with blanking and sync signals from the TV Sync Generator to produce a composite video signal which will drive the TV monitor. Three adjustments are provided:

1) Sync level
2) Blanking level
3) Video Gain

As an auxiliary function, this card also accepts the composite synchronization signal from the Fairchild 3262A and after buffering provides it to the TV camera as genlock sync in order to keep the camera in step with the rest of the system.

SECTION III

TV CAMERA BUS INTERFACE CARD

This device is a plug-in Quad-wide card to the PDP-11 UNIBUS which will accept digitized TV data (picture elements or pixels) from a high-speed Analog-to-Digital Converter (ADC), buffer them in a FIFO memory, and as a Slave Device transfer them to a UNIBUS Master -- typically a Frame Store Memory or Microprocessor. It has two modes of operation: Snapshot and Stripe. In the first mode, it sends an entire field to the Frame Store Memory. In theory this could consist of 256 TV lines, each of 256 pixels. Words are packed, however, so that two 8-bit pixels are contained within a single PDP-11 16-bit word. In the Stripe mode, the TV BIC sends only one eighth of each line, consisting of 32 pixels, to the microprocessor. This vertical piece of the TV picture is referred to as a Stripe.

3.1 Block Diagram

Data flow in Fig. 10. is from left to right, from the ADC through the Hold Register, then the 8-bit wide FIFO memory, and finally to the UNIBUS as shown on the right hand side. Note that the 8-bit Data Register permits assembling two pixels into a single word. Also, the 8-bit Control and Status Register's contents can be multiplexed onto the UNIBUS when commanded by the computer. Function control and I/O control logic responds to bus commands to carry out the required timing and sequencing.

In order to control the ADC, the TV BIC must send Start Conversion. The ADC responds with an 8-bit data input and the signal, End of Conversion. This can be used to strobe data into the 64-word FIFO.

Also on the left side of the figure are signals received from the System Synchronization Unit. These include the square-wave 4.9 MHz PCLK (pixel clock) needed to drive the ADC plus signals from the Fairchild 3262A TV sychronization chip. The latter include Vertical Drive, Composite Blanking, Even Field, and Odd Field.

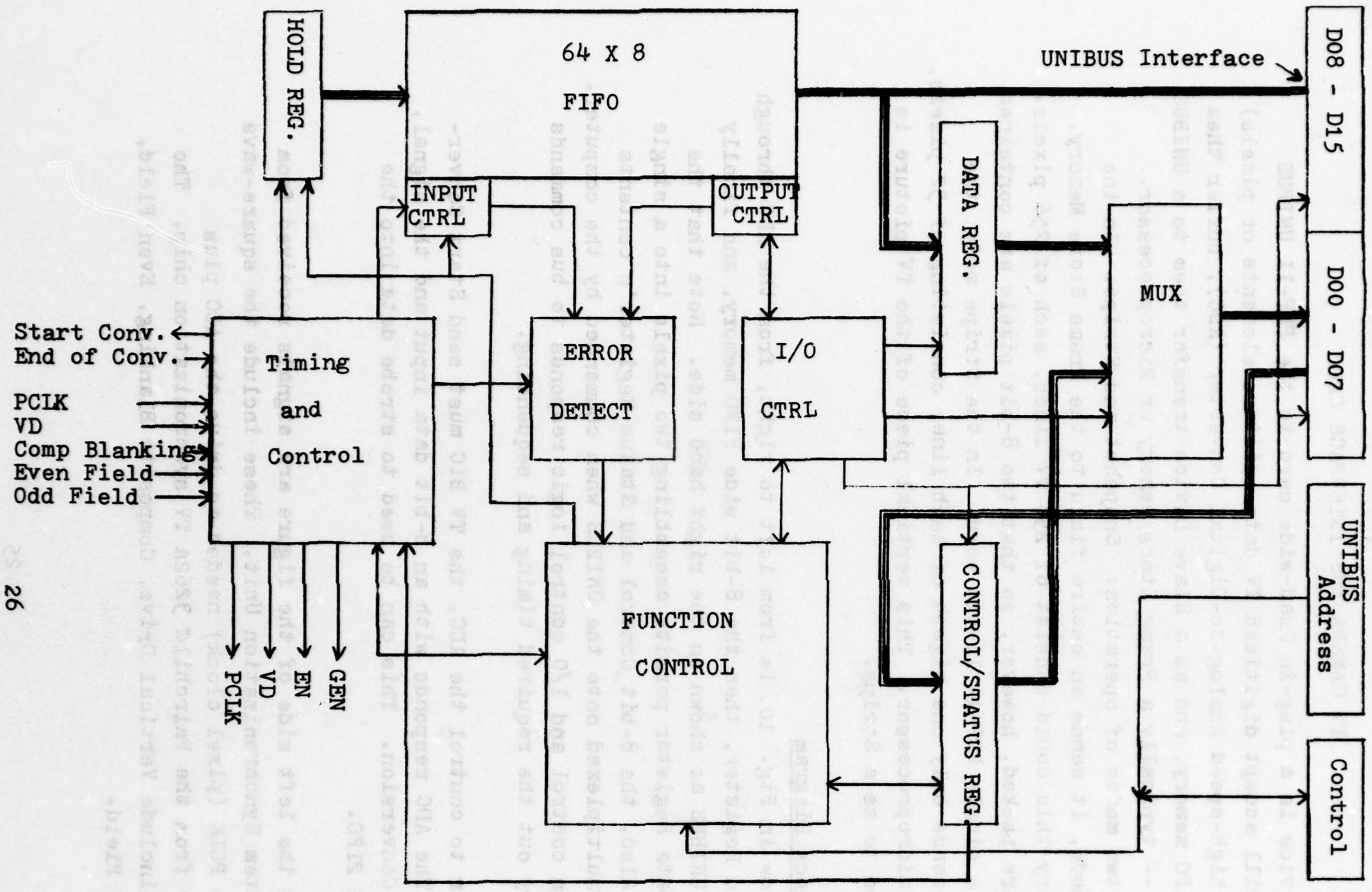


Figure 10. TV Camera BIC Block Diagram

Signals sent to the Frame Store Memory BIC include PCLK, Vertical Drive, Enable, and Gate Enable. The last three of these are sent also to the microprocessor BIC. These signals can be wired along the backplane of the PDP-11 since they are sent between Bus Interface Cards plugged into the UNIBUS.

PCLK is used by the FSM BIC for reading pixels from the Frame Store Memory to update the TV Monitor and for other timing functions. In particular, the FSM BIC divides it down to CCLK, which corresponds to a 2-byte word time. There are 312 PCLK clocks during a TV line -- 256 during active time and 56 during retrace. Because the PCLK, which is counted down from a higher frequency crystal oscillator, is closer to 4.914 MHz than to 4.9 MHz, the active line time is 52.1 usec rather than the usual 52.2. Retrace time, which consists of 56 clocks, is then 11.4 usec rather than the usual 11.3 usec.

Vertical Drive indicates that the camera is generating data corresponding to the active 485 lines of a picture. When Vertical Drive goes false, it means that vertical retrace has begun. Since the fields are interlaced (even and odd) to form the picture, each field has approximately 242½ active lines and 20 lines for retrace. The retrace time is thus about 1.27 msec.

Enable (EN) indicates the active portion of the TV line. This portion, as indicated above, contains 256 PCLK clocks, and there are 56 PCLK clocks during the negation of EN. This signal is needed both by the Frame Store Memory and by the microprocessor. In a sense they are sychronized to the camera and to the monitor.

Gate Enable (GEN) is a signal which indicates that under control of the TV BIC the ADC is converting analog pixels to digital values. This signal is useful to the other BIC's since it indicates that there is data in the FIFO ready to be transferred out by a Master device.

3.2 Operation

When placed in the snapshot mode by the PDP-11 computer, the TV BIC will capture an entire field of approximately 242½ lines. It has been designed so that it will always select an odd frame. This is convenient because it avoids the half line which starts the even field. Also the half line at the end of the odd field does not create a problem because it happens during vertical retrace. Because the even and odd fields are different (interleaved), it is better to always take data from one or the other when, as in the present case, not all data can be used. Otherwise the eye can detect differences. These differences occur when first an odd field is "doubled up" to 525 lines to make a frame and later an even field is used in a similar manner.

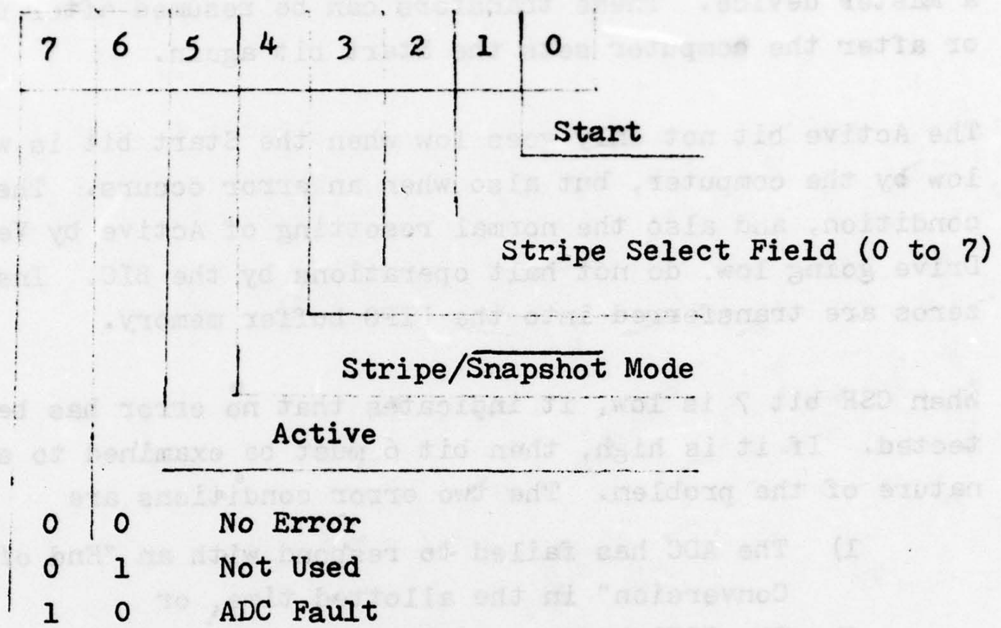
A 64 word buffer is sufficient because the UNIBUS and the Frame Store Memory are able to keep up with the TV camera in terms of data rate -- 4.9 megapixels per second peak rate. Note, however, that by continuing to transfer data during the retrace time the average data rate can be reduced by the ratio of active line time to total line time, or 52.1/63.5 = .82. .82x4.9 = 4.02 megapixels per second average rate, which is equivalent to a word rate slightly in excess of 2 MHz.

When the vertical drive signal goes false, indicating that vertical retrace has begun, the TV BIC will start supplying zeros to be stored in the FIFO. This has the effect of clearing the FIFO until a new field transfer is begun at a later time.

In the stripe mode which is used when a microprocessor is going to process data directly from the TV camera without going through a FSM, 32 pixels from each line are selected and stored in the FIFO buffer. The stripes are numbered 0 to 7 from left to right. The stripe currently being read is determined by 3 selection bits in the Control and Status Register. The resulting octals numbers, 0, 2, 4, and 6, will select the odd field, while 1, 3, 5, and 7 select the even field. When selecting the even field, the TV BIC is careful to disregard the first half line at the top of the field. When the vertical drive signal goes false, zeros are again transferred into the FIFO.

3.3 Control and Status Register

The CSR in the TV BIC is organized as follows:



0	0	No Error
0	1	Not Used
1	0	ADC Fault
1	1	Overflow Error

The computer must write a one into CSR0 in order to initiate the capture and storage of pixels. The Stripe Select Field has already been discussed, as has the stripe/snapshot mode.

When the Start bit is set by the computer, the Active bit is automatically raised by the BIC. The next odd or even field pulse (whichever is appropriate in the selected Mode and Field) will reset the Start bit. This occurs about 200 usec before viewable data actually is available from the TV camera and ADC. The Active bit, however, normally stays high until vertical drive goes false. Thus the Active bit is a status bit which the computer can read. If the computer writes the Start bit low, it will cause the Active bit to also go low. When the Start bit is thus reset, the effect is a general reset. In addition, the UNIBUS INIT has the same effect. Either of these sets the Reset F/F, which will be cleared by "unblank", the negation of Composite Blanking. While the Reset F/F is high, it prevents Data Register transfers along the UNIBUS in response to a Master device. These transfers can be resumed after unblank or after the computer sets the Start bit again.

The Active bit not only goes low when the Start bit is written low by the computer, but also when an error occurs. The error condition, and also the normal resetting of Active by Vertical Drive going low, do not halt operations by the BIC. Instead zeros are transferred into the FIFO buffer memory.

When CSR bit 7 is low, it indicates that no error has been detected. If it is high, then bit 6 must be examined to see the nature of the problem. The two error conditions are

1) The ADC has failed to respond with an "End of Conversion" in the allotted time, or
2) The FIFO buffer was full and an attempt was made by the TV BIC to write into it.

SECTION IV

FRAME STORE MEMORY AND ITS BUS INTERFACE CARD

Because the updating of the TV image from the microprocessor does not occur at full TV rates, it is necessary to carry out a scan converter function. This is done digitally by means of a 256x256 = 65,000 byte semiconductor memory, referred to as the Frame Store Memory (FSM). In order to keep the display on the TV monitor refreshed, 256 pixels along each TV line must be accessed from RAM, converted from digital to analog, and combined with a sync signal to produce composite video. Even though two pixels are stored in each RAM location -- the FSM being organized as 32,000 16-bit words -- the required speed of operation is 128 words in an active line time of 52.2 microseconds, which is equivalent to 2.45 MHz. This in turn requires a memory cycle time of 407 nanoseconds or less.

On the one hand, the FSM must drive the TV monitor and keep it refreshed from its stored pixel values, each an eight-bit byte. On the other hand, the FSM must acquire the data to be displayed. This is accomplished by means of a Bus Interface Card which ties the FSM to the PDP-11 UNIBUS. The BIC contains all the necessary logic to permit the FSM to appear either as a Master or Slave device on the UNIBUS. Three other types of devices normally store data in the FSM or read from it. They are the

a) TV Camera
b) microprocessor
c) PDP-11 computer

In the Master Mode, the FSM can capture an entire field of data (256 lines of 256 pixels each from the Camera, whose own BIC will buffer the digitized samples from the high-speed A to D converter. In the Slave Mode the FSM can be loaded with data from the microprocessor or even the PDP-11. Similarly it can be read by the latter two devices. the FSM is made to appear to the computer as an extension of PDP-11 core memory. For simplicity the FSM is divided into stripes just as the TV picture is. The vertical stripes are each 32 pixels wide and 256 lines deep. This amounts to 8,192 pixels or 4,096 words. A stripe register in the BIC set by the computer keeps track of which stripe is currently "active" for purposes of UNIBUS transactions. It is assumed that the PDP-11 normally does not have a full 28k-word core memory. If not, then the 4k-word stripe of FSM will be mapped into the unused portion of 28k core memory space. This has the advantage that the PDP-11 can now directly access the FSM for data manipulation or data analysis. The FSM can be made to appear as "fast" memory.

With the exception of the PDP-11 CPU all other devices on the UNIBUS make use of an 18-bit address. It is therefore convenient to give the FSM two addresses -- one within the normal 28k memory space and one in "upper" memory which must be accessed with an 18-bit address. The latter is convenient because it permits having two or more FSM's on the UNIBUS at once. They are assigned different addresses in upper memory to avoid confusion. If possible, the two FSM's should be given different 4k address ranges in the 28k of lower memory. The PDP-11 is able to set bits within the CSR of each FSM BIC designating which is currently active for reading and which for writing. However, the PDP-11 always precedes a write command with a read. Therefore, to do a write from the PDP-11, the read and write bits of the Addressed FSM must be set and the same bits must be cleared in the inactive FSM BIC CSR prior to the write operation. In this way, more than one FSM may occupy the same address range.

4.1 Frame Store Memory

The Frame Store Memory is the Microram 3400N of Electronic Memories & Magnetics Corp. The particular model selected is a high-performance version with a cycle time of less than 400 nsec rather than the 450 nsec shown in the EM&M reference document included with this report. The EM&M Technical Manual, TM928776 Rev B of January, 1976, presents the theory of operation of the memory, and only certain extracts and highlights from this document will be included here.

The basic memory consists of NMOS chips packaged on a single card, shown in Fig. 4-1. Each chip holds 4k bits so that the maximum size memory (32,768 words of 18 bits each) requires 144 chips. Data/Ware packages this card within a suitable chassis equipped with fans. Power is supplied from external

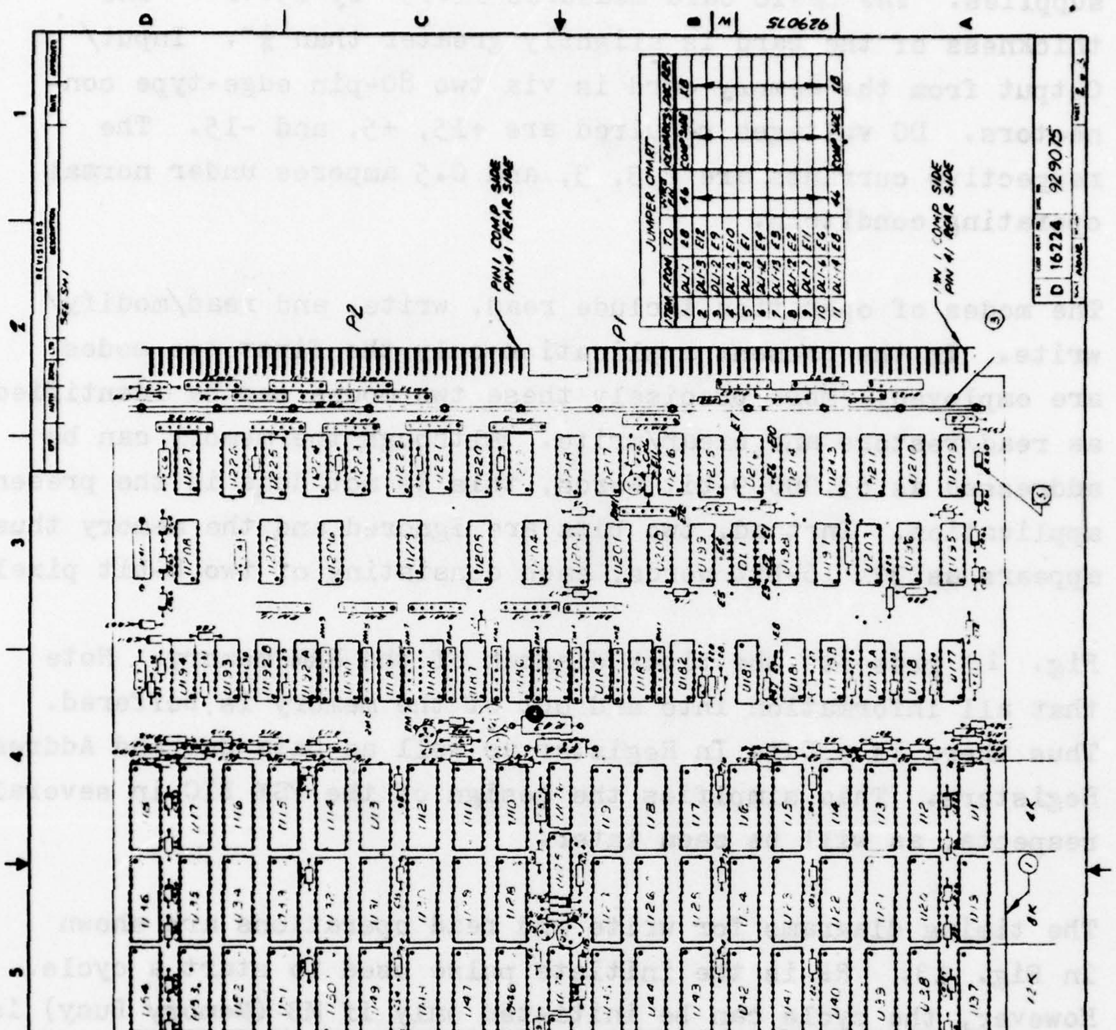


Figure 11. Card Layout

supplies. The basic card measures 11.75" by 15.40". The thickness of the card is slightly greater than ½". Input/Output from the memory card is via two 80-pin edge-type connectors. DC voltages required are +15, +5, and -15. The respective currents are 2.3, 3, and 0.5 amperes under normal operating conditons.

The modes of operation include read, write, and read/modify/write. In the present application only the first two modes are employed. More precisely these two modes can be identified as read/restore and clear/write. Although the memory can be addressed as 65,000 9-bit words, this is not done in the present application. Instead, two bits are ignored and the memory thus appears as 32k 16-bit words, each consisting of two 8-bit pixels.

Fig. 12 presents the block diagram of the RAM memory. Note that all information into and out of the memory is buffered. Thus there is a Data In Register as well as Data Out and Address Registers. This simplfies the design of the FSM BIC in several respects, as will be seen later.

The timing diagrams for write and read operations are shown in Fig. 13. RP is the initiate pulse used to start a cycle. However, the cycle can be initiated only if MB (Memory Busy) is high indicating that the memory is <u>not</u> busy. The write timing diagram shows that the RP line must have been high at least 50 nsec before going low and must produce a pulse of at least 50 nsec width. The AI (Address In) and DI (Data In) lines must be stable at the time the RP pulse goes low.

When reading, Fig. 13 shows that DA (Data Available) drops after 275 nanoseconds to indicate that DO (Data Out) is ready. In the present application this is a valuable signal when the FSM is refreshing the TV monitor. It has been noted that when this is being done, words must be read from the FSM at a rate of one word every 407 nsec, which is challenging.

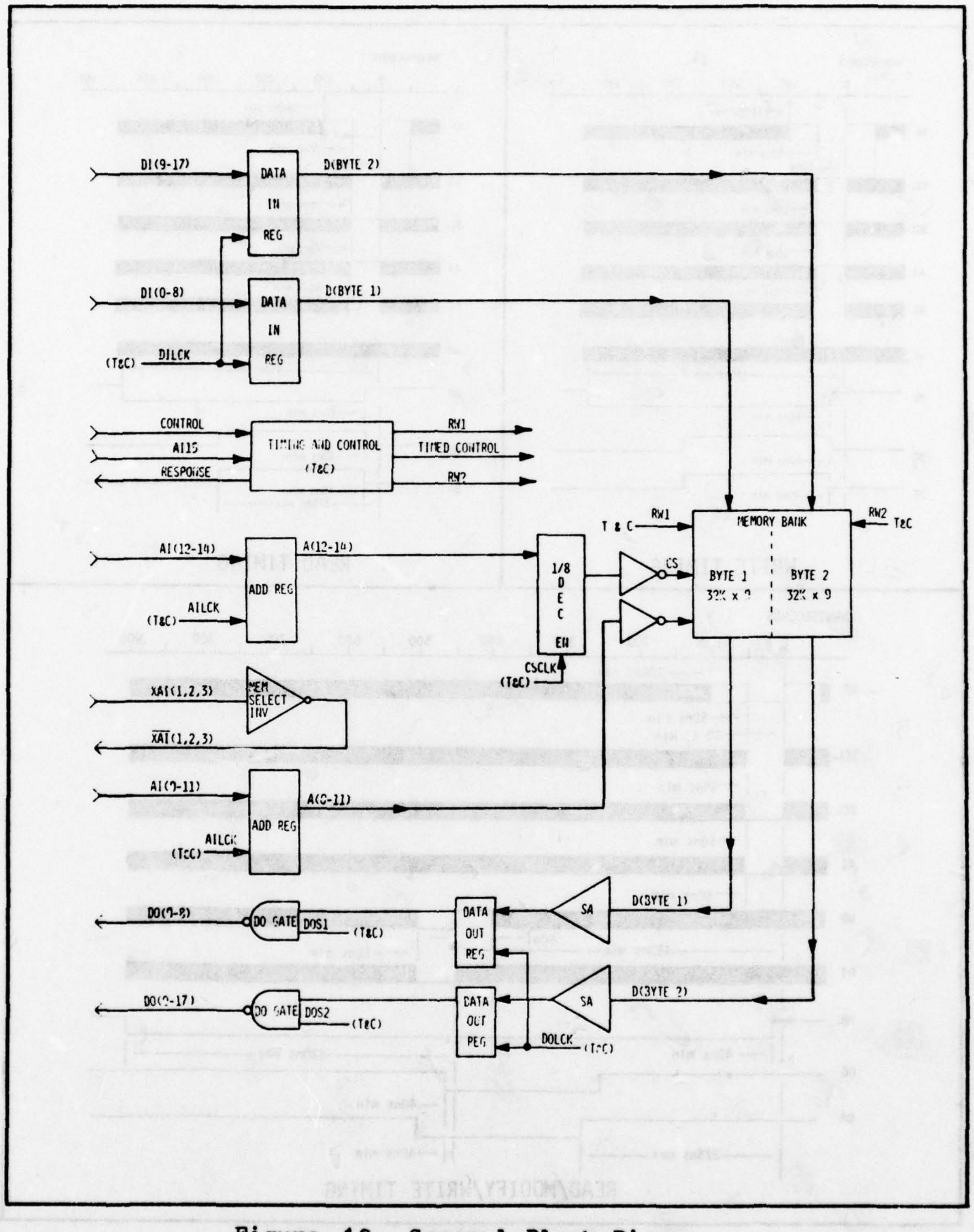


Figure 12. General Block Diagram

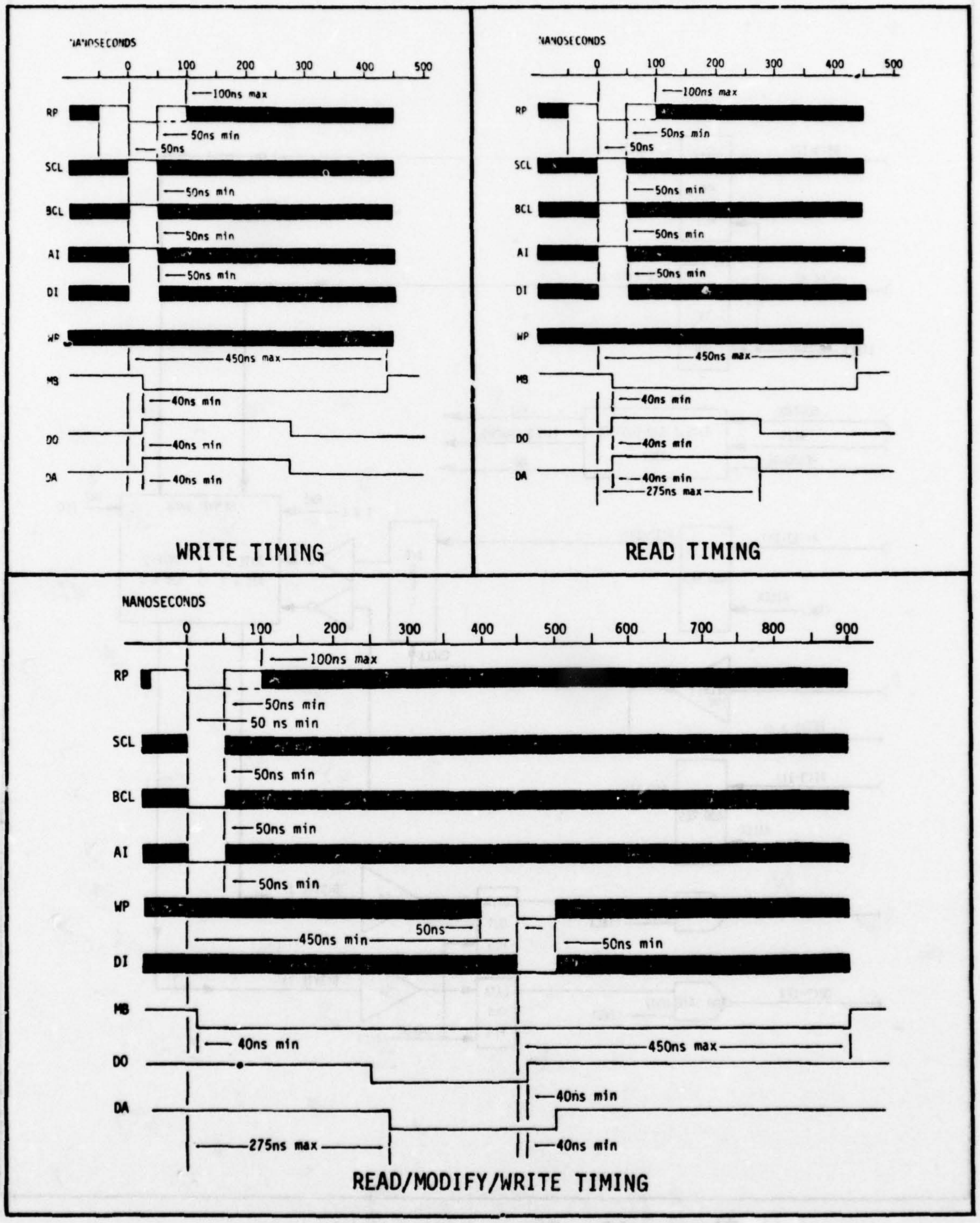


Figure 13. Interface Timing Diagram

4.2 Bus Interface Card Characteristics

Fig. 14. is the block diagram of the FSM BIC, which is a full-sized Hex board which plugs directly into the PDP-11 chassis and is powered from the computer power supply. In the upper right hand corner of the diagram it can be seen that it has the necessary logic to act either as Bus Master or Bus Slave, as directed by the computer. Suitable drivers and receivers are provided for the data and address lines. In order to act as Bus Master, the BIC must supply an address, in particular that of the TV Camera. As mentioned above, the FSM can be addressed as either a 4k part of lower or upper computer memory. Thus an address comparator is needed, as shown. So that the computer can write and read the Control and Status Register, it is necessary to decode this address on the UNIBUS.

Both a Write Register and a Read Register are provided in order to buffer data from and to the UNIBUS. These registers communicate with the FSM. So that the FSM can refresh the TV monitor, an 8-bit buffer is set aside solely for this function. But a multiplexer is needed to select first one byte and then the other from the 16-bit word from the FSM. The multiplexer in turn drives the digital-to-analog converter, which is not part of the FSM BIC.

The address supplied to the FSM comes from the Scan Counter during the Master Mode. In the Master Mode words are read from the TV Camera BIC over the UNIBUS until vertical retrace occurs. External synchronizing circuitry assures that the FSM BIC starts the transfer of a complete field at the appropriate moment and stops at vertical retrace. Actually fewer than 256 lines are transferred. The words are stored sequentially in the FSM.

In the Slave Mode, the address also comes from the Scan Counter normally. The exception is when some other device acting as

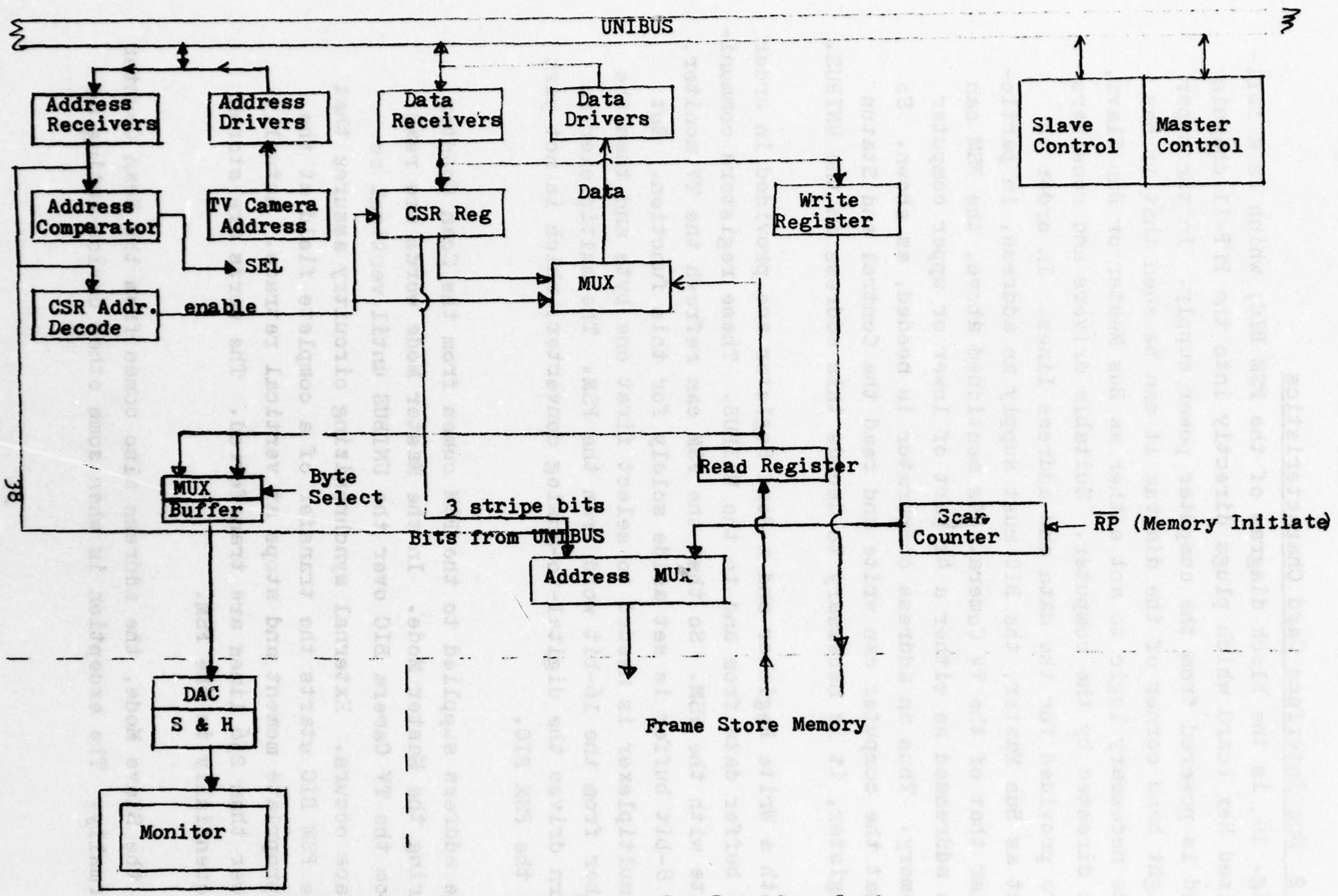


Figure 14. Frame Store Memory Bus Interface Card

Bus master elects to read or write a word in the FSM. When this occurs, a "cycle steal" suspends the updating of the TV monitor for a single memory cycle. Afterward, the Scan Counter again supplies the address which follows, just as the one preceding this stolen address. During horizontal retrace, the Scan Counter does not advance in the Slave Mode, so as not to get out of step with the sweep circuitry in the TV Monitor. Note that the 3 "stripe" bits of the "UNIBUS address" come from the CSR, where they were set by the computer. These 3 bits determine which "stripe" of 4k words is currently accessible to the devices (including the computer) on the UNIBUS.

When the computer or the microprocessor address the FSM, an interesting distinction must be made between read and write operations. Writing is easier in that both the address and data are supplied. Then the Master can quickly give up the bus and let the FSM -- once it has the address and data in its buffers -- carry through the cycle. But when reading the Master must supply the address and then wait at least until the FSM signals Data Available before the transaction can be completed. In order to speed up the latter, a "fast mode" has been added during FSM Slave operations. When the computer sets a suitable bit in the CSR, any device on the UNIBUS reading from the FSM is given the current contents of the Read Register as soon as the address is supplied. These contents correspond to the address previously supplied. The contents of the current address will be fetched and will be ready during the next UNIBUS read from the FSM. Thus if the computer or microprocessor is reading a long string of words from FSM, this mode will speed up the process. The only cost is one extra cycle and some bookkeeping by the computer. In the fast mode there is no refresh.

4.3 Master Mode

As mentioned the FSM BIC will transfer words from the TV Camera by supplying a constant address on the UNIBUS and by advancing the Scan Counter, which is initially reset, after each bus transaction. It should be noted that the Scan Counter is not advanced by a clock but rather it is advanced only when a word is received from the TV Camera. Thus the unique self-timing or hand-shaking feature of the UNIBUS is employed. This means the use of the Master Sync signal sent by the Master Device (in this case the FSM) and the Slave Sync signal returned by the Slave Device when it has placed the desired word on the UNIBUS.

The camera uses a FIFO buffer and so it is not necessary to give it a word address. The FSM supplies a constant address to the TV Camera BIC, and each time it does so it receives a word in return. This makes it possible to transfer the 128 words corresponding to a full line in 63.5 microseconds instead of the active line time of 52.2 microseconds. This reduces the data rate from 2.45 MHz to 2.0 MHz, which is very desirable in order to stay within the capabilities of the UNIBUS.

Several methods of employing the system are possible. The Master Mode permits capturing an entire field of TV data for later processing. The sync system notifies both the TV Camera BIC and the FSM BIC of the start of a new field. The FSM BIC will not request the bus until a signal "Gate Enable" from the TV Camera BIC informs it that the data is ready in the buffer for transfer. When an entire field is being transferred, as in this mode, it is always the odd field selected. When successive fields are captured, processed, and displayed on the monitor, this results in a higher quality picture since the even and odd fields are not intermixed, which would create artifacts. The vertical retrace stops the transfer of data from the camera and also removes the FSM BIC from the Master Mode.

While in the Master Mode, the FSM BIC stops refreshing of the display. As soon as the BIC returns to the Slave Mode, the refreshing is resumed.

4.3.1 Timing

The timing for the bus transfers in the Master Mode is shown in Fig. 15. The computer initiates this mode by setting CSR bit 0 to a logical one. When the TV camera BIC card sends $\overline{\text{GEN}}$ (Gate Enable) indicating that it has started the acquisition of data from an odd field (upper, left hand corner of the TV screen), CSR bit zero is strobed into a latch and the Master Mode begins. When bus control is granted, MRES (Master Reset) zeros the counter in the BIC which will step sequentially through the FSM addresses as the field of data is stored. At the same time bus control is granted to the BIC, a one-shot is triggered to measure off a time interval of about 300 micro-seconds. If the camera interface does not respond with SSYN to the FSM BIC's MSYN along the UNIBUS, an error bit (bit 7) is set in the CSR.

After being granted control of the bus, the TV camera address is placed on the UNIBUS. Following a 150 nanosec. delay to eliminate skewing problems, MSYN (Master Synchronization) is raised by the FSM BIC to request information from the camera. The camera responds with SSYN (Slave Synchronization) as part of the hand-shaking. Since it is the responsibility of the Master device to eliminate skew, the BIC will delay SSYN by 75 nsec -- creating delayed SSYN, designated as DSSYN. DSSYN strobes data into the Write Register and causes MSYN to be dropped. When the TV camera sees MSYN dropped, then it will in turn drop SSYN to complete the bus transaction. (On the logic schematics, SSYN is identified as UDSSYN for undelayed SSYN.)

In order to initiate the FSM write cycle, two conditions must be met. The first is that the TV data has been acquired by the FSM BIC the second is that the memory has completed the previous cycle. Suitable logic involving the RP F/F triggers the RP one-shot after SSYN has fallen.

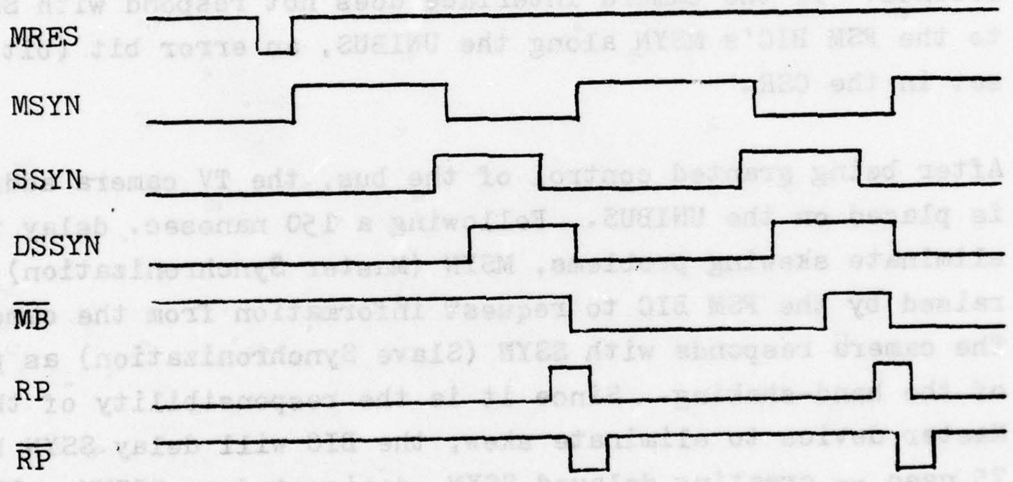


Figure 15. Master Mode Timing

4.3.2 Implementation

In simplified form, Figure 16. shows the generation of the timing signals of Figure 15. At start-up of the Master Mode, the rising edge of the 150 nsec MRES pulse sets the MSYN F/F, which generates MMSYN. The first bus transaction after the Master Mode is entered requires that MSYN be delayed 150 nsec for deskewing. However, thereafter this is not necessary since the address is left on the bus and is unchanged. This speeds up the bus transfers.

After MSYN is put on the bus and the TV camera BIC responds with SSYN, the leading edge of the latter signal, qualified by MASTER and MSYN, NORRED with CABUS -- resets the MSYN F/F. As soon as the TV camera BIC responds by dropping SSYN, this trailing edge of the undelayed signal ($\overline{\text{UDSSYN}}$) sets the RP F/F to request a memory cycle. This is appropriate because the UNIBUS data was strobed into the Write Register on the leading edge of SSYN and is therefore available.

Since the memory is faster than the number of words converted per line, there is no danger of the CAMERA BIC FIFO getting ahead of the FSM. Therefore, it is safe to use the rising edge of the Q output of the RP F/F to trigger the RP one-shot, which in turn produces a 75 nsec RP, as required by the FSM. $\overline{\text{RP}}$ then clears the RP F/F.

From the logic diagram it can be seen that $\overline{\text{RP}}$ is also used to advance the Scan Counter, whose output is the address where the next input word will be stored. In order to start the fetch from the TV camera of the next word, the low going RPNOT pulse is used to preset the MSYN F/F.

When MSYN rises, the TV camera is being requested to send another word. Note that this occurs before or at most slightly after the FSM start to write the current word into the current address, thus taking advantage of the fact the FSM has internal registers.

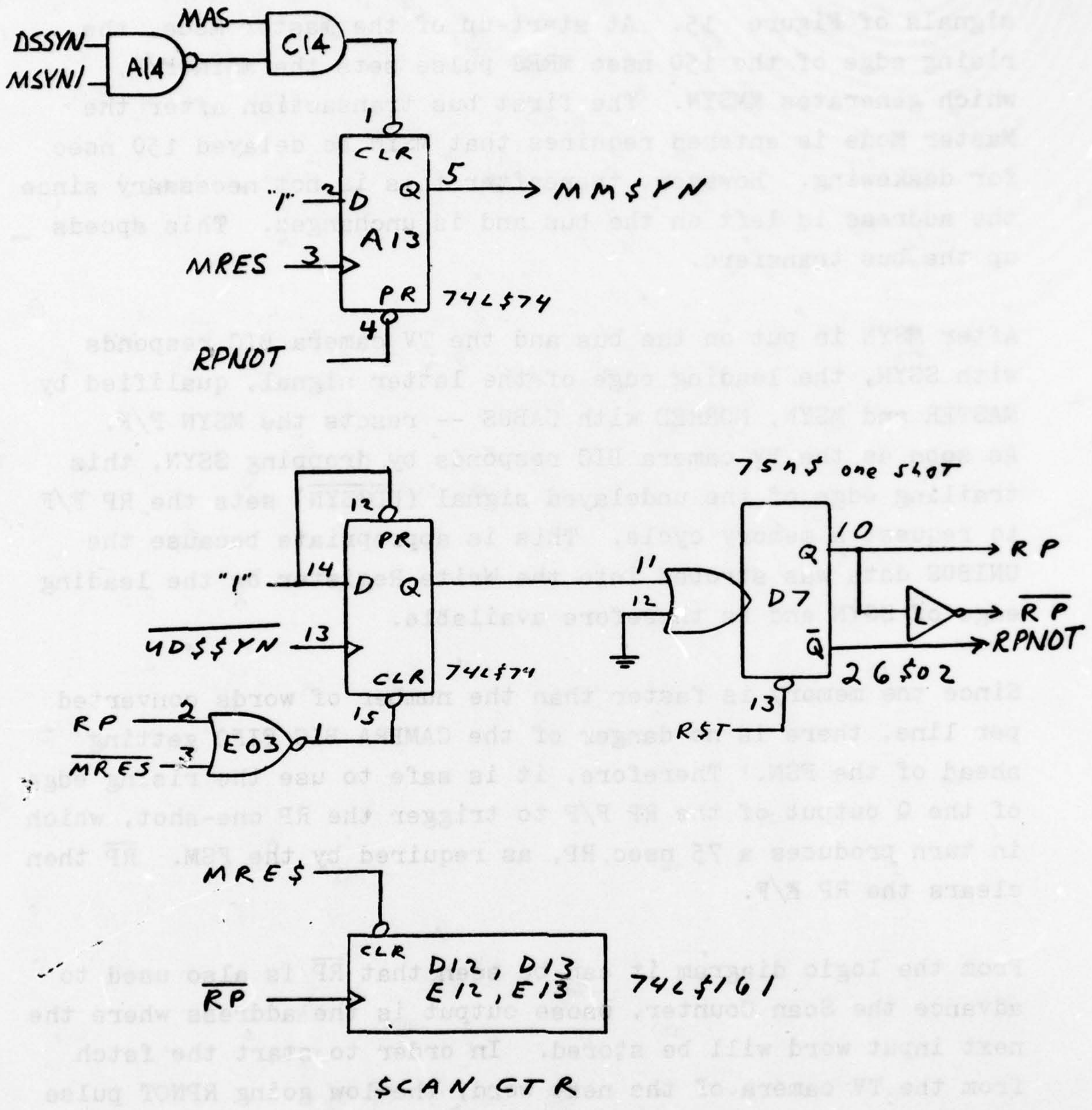


Figure 16. Master Mode Logic

4.4 Slave Mode

In the Slave Mode the FSM BIC must permit cycle steals from the UNIBUS while continuing the refresh of the Monitor. This requires care in design because of the very short cycle times involved. Furthermore, when a cycle steal occurs, this means that data normally made available from the FSM to the Monitor is lost. This is one reason why the Sample/Hold Enable signal is included. It makes it possible to "Extrapolate" the last pixel value over the next two pixel locations. If the normal display includes Byte 6, Byte 7, Byte 8, Byte 9, Byte 10, Byte 11, ..., and if Bytes 8 and 9 are not fetched from FSM because of a cycle steal, then it is Byte 7 which should be extrapolated into the positions of Byte 8 and 9 as a substitute. This problem does not arise if the cycle stealing occurs during horizontal refresh -- a period of time of some 11.3 microseconds duration. The heaviest demand on the FSM is from the microprocessor and so the logic in its BIC has been designed to force it to write to the FSM during retrace time. In a two microprocessor system, this is all that is required. However, if a single microprocessor is accessing FSM both to read out pixels and then to write them back after processing, it must be given access during active line time. This is when the cycle stealing is required.

4.4.1 Refreshing TV Monitor

In the Slave Mode the most important function of the FSM BIC is the sequential access of 128 words (256 bytes) during the 52.2 microseconds of active line time. These bytes are converted to analog form and displayed on the Monitor.

The logic is shown in Fig. 4-7. The Scan Counter is split into two parts. The seven least significant bits count up the 128 words and come from the Pixel Counter. The eight most significant bits come from the Line Counter. The Scan Counter is advanced only during active line time by CCLK·PCLK-- pulses and is cleared at Vertical Retrace time.

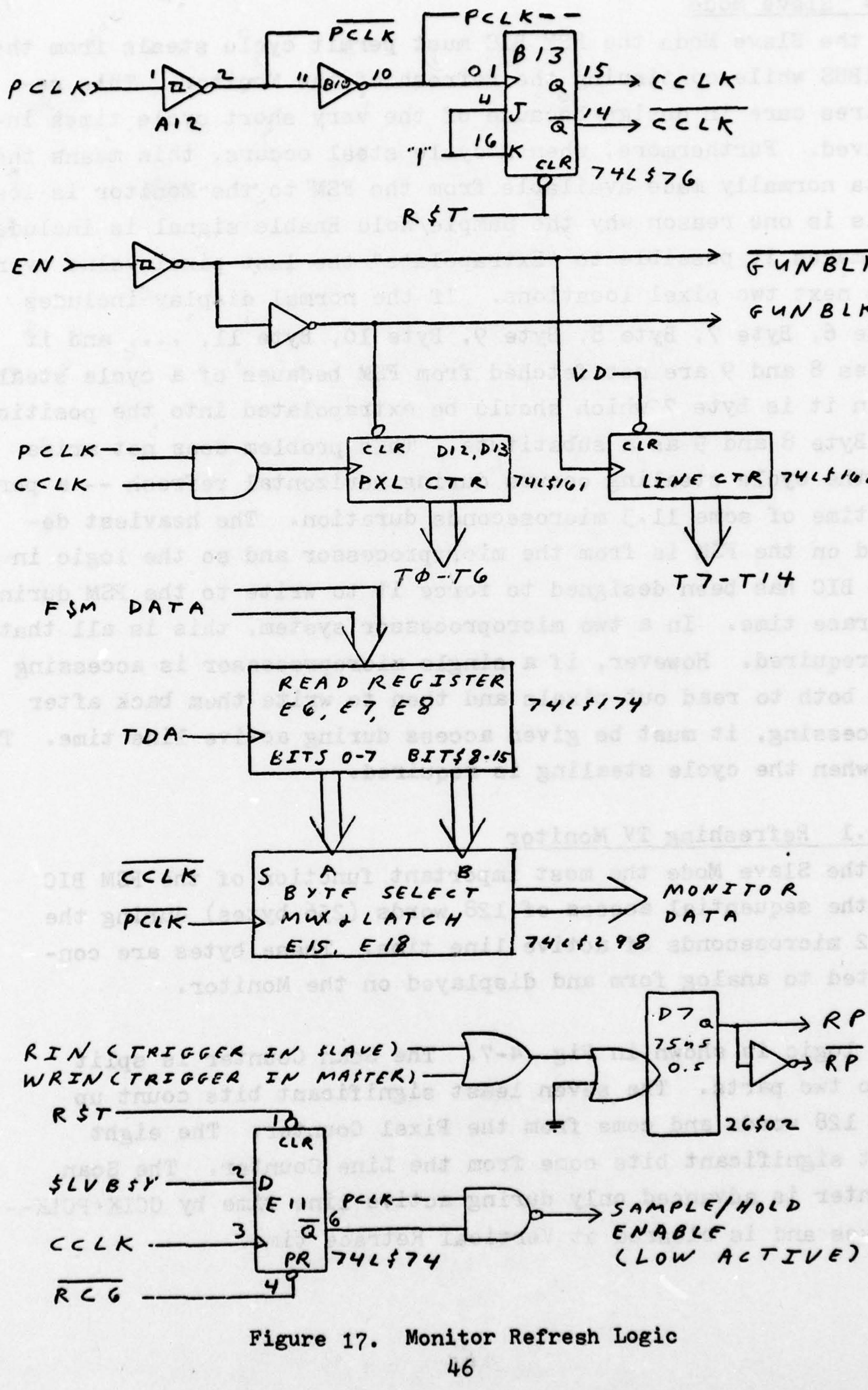


Figure 17. Monitor Refresh Logic

Basic timing is furnished by the Sync System and consists of a 4.9 MHz clock, the PCLK or Pixel Clock. Through a flip-flop, PCLK is divided by two to produce a clock, CCLK, which is a word-time clock. A signal, Enable or EN, from the camera interface clears the pixel counter at the end of each line while Vertical Drive clears the line counter at the end of a field. EN is inverted and renamed Gated Unblanking or $\overline{\text{GUNBLK}}$, which is equivalent to horizontal retrace time.

Fig. 17. shows the Line and Pixel Counters, the flip-flop for counting down the PCLK to the CCLK, the one-shot for producing the memory initiate pulse, RP, the Read Register which holds data to be sent to the D/A converter when the FSM indicates Data Available (DA), and the Byte Select Mux which selects the alternate 8-bit bytes from the 16-bit memory word. During an active line time CCLK generates RP to initiate memory cycles. The rising edge of CCLK ANDed with PCLK--will advance the Scan Counter so that the next address is ready well in advance. When CCLK is high the high-order bits of the Read Register are selected; when CCLK is low the low-order bits are selected. The sample and hold module is enabled by $\overline{\text{PCLK}}$ so that the output of the DAC has more than 100 nsec to stabilize before its output is latched.

The timing for the Slave Mode is shown in Fig. 18. The duration of the positive-going portion of PCLK is approximately 100 ns. Thus the complete cycle of CCLK, which corresponds to a cycle of the Frame Store Memory, is about 400 ns. K clock is $\overline{\text{CCLK}}$·PCLK--and is useful for initiating switching actions and making decisions prior to the leading edge of CCLK -- the latter being the basic clock for timing. Note that in the Slave Mode, during active line time but not during retrace, the leading edge of CCLK triggers the memory with the memory initiate pulse, RP. The latter pulse makes the FSM generate MB, i.e. Memory Busy. Memory Not Busy ($\overline{\text{MB}}$) must rise before the next cycle. This means that the FSM must cycle in 400 nsec or less. When reading, the signal Data Available, or DA, indicates that the data is ready to be strobed into the Read Register in the BIC, but the read register is actually strobed by KCLK about 20 ns later, during active line time.

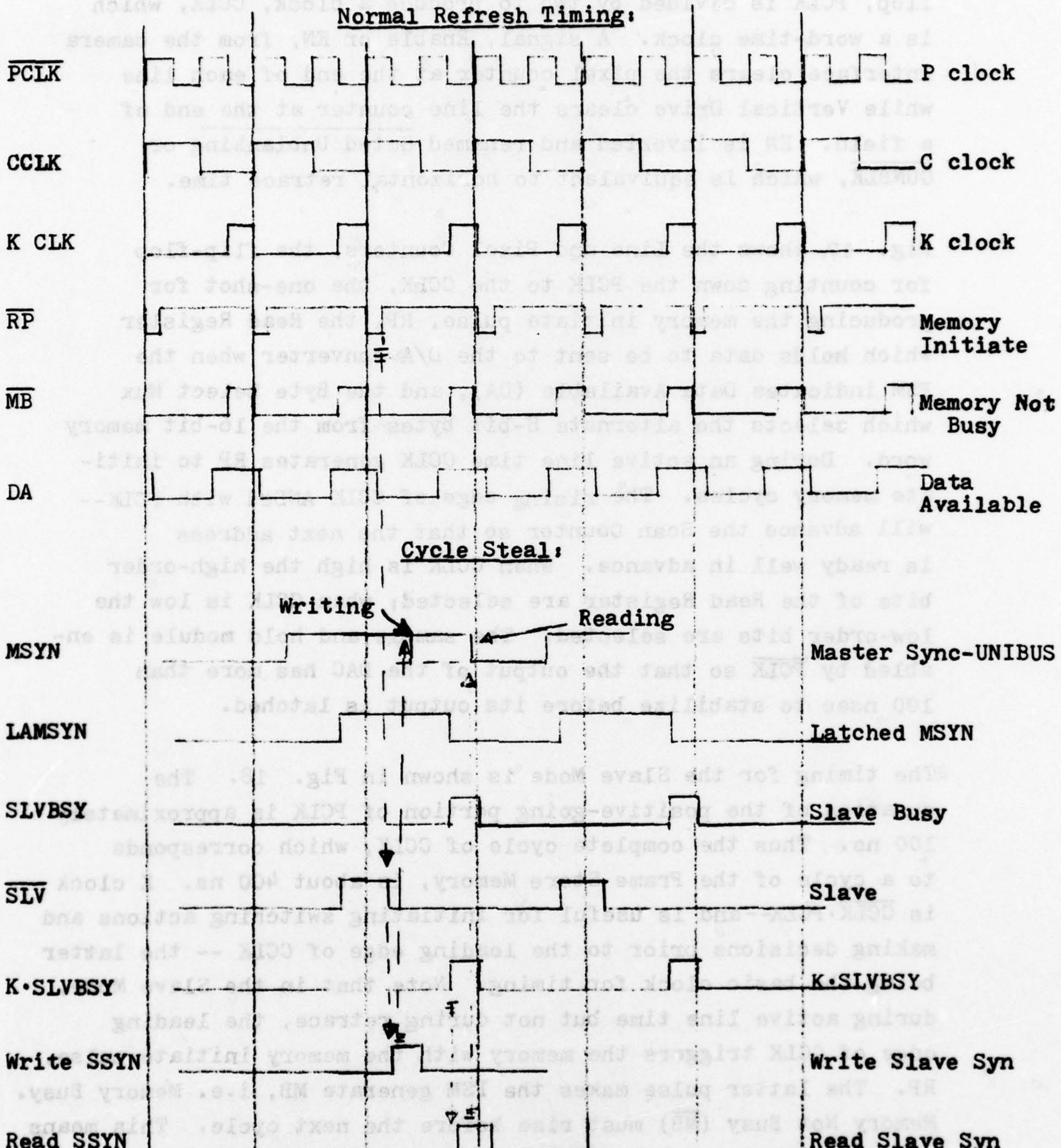


Fig. 18. Slave Mode Timing

Continuing the discussion of active line time, it is desired to permit devices on the UNIBUS, such as the PDP-11 computer itself or the microprocessor, to steal a cycle in order to read or to write memory. The lower part of Figure 18. depicts how this can be done.

Assume that MSYN goes high and assume furthermore that the address on the UNIBUS indicates that the FSM is the SLAVE. Then an internal signal, DEVADD, indicates that this is the case. The K clock is the synchronizing signal to decide whether DEVADD is or is not to be recognized. K clock is convenient since it precedes the rising edge of CCLK and the memory initiate by 100 nsec. From Figure 19, it can be seen that if DEVADD is present and if it is not retrace time, then K clock ($\overline{CCLK}$•PCLK--) will set the latched LAMSYN F/F. Also, the next K clock will reset LAMSYN and will simultaneously set Slave Busy (SLVBSY), also for exactly one cycle. As LAMSYN is being raised to the logical one state, its output is also setting the Slave F/F so that $\overline{SLV}$ is taken to a high state, which multiplexes the external address onto the FSM Address Bus. After $\overline{SLV}$ is in the high state, the memory initiate pulse, $\overline{RP}$, will reset $\overline{SLV}$. Thus $\overline{SLV}$ is a rather short pulse which rises with the leading edge of LAMSYN, i.e. during a memory cycle steal, and falls shortly after the memory cycle is initiated, being stable slightly before till slightly after RP. The main use of $\overline{SLV}$ is during the UNIBUS write because it is a convenient way to set the answering signal, SSYN, required by the UNIBUS protocol in response to MSYN from the Master Device. SSYN says that the data has been received by the FSM and that the transaction is complete. This is not true since the data is in the process of being written into the FSM, but by releasing the UNIBUS early data transactions are speeded up.

Note, however, that LAMSYN can only be set on every other cycle of the FSM during active line time, as depicted in Figure 18. On the first K clock it is set, on the second it it reset, on the third it could be set again as depicted.

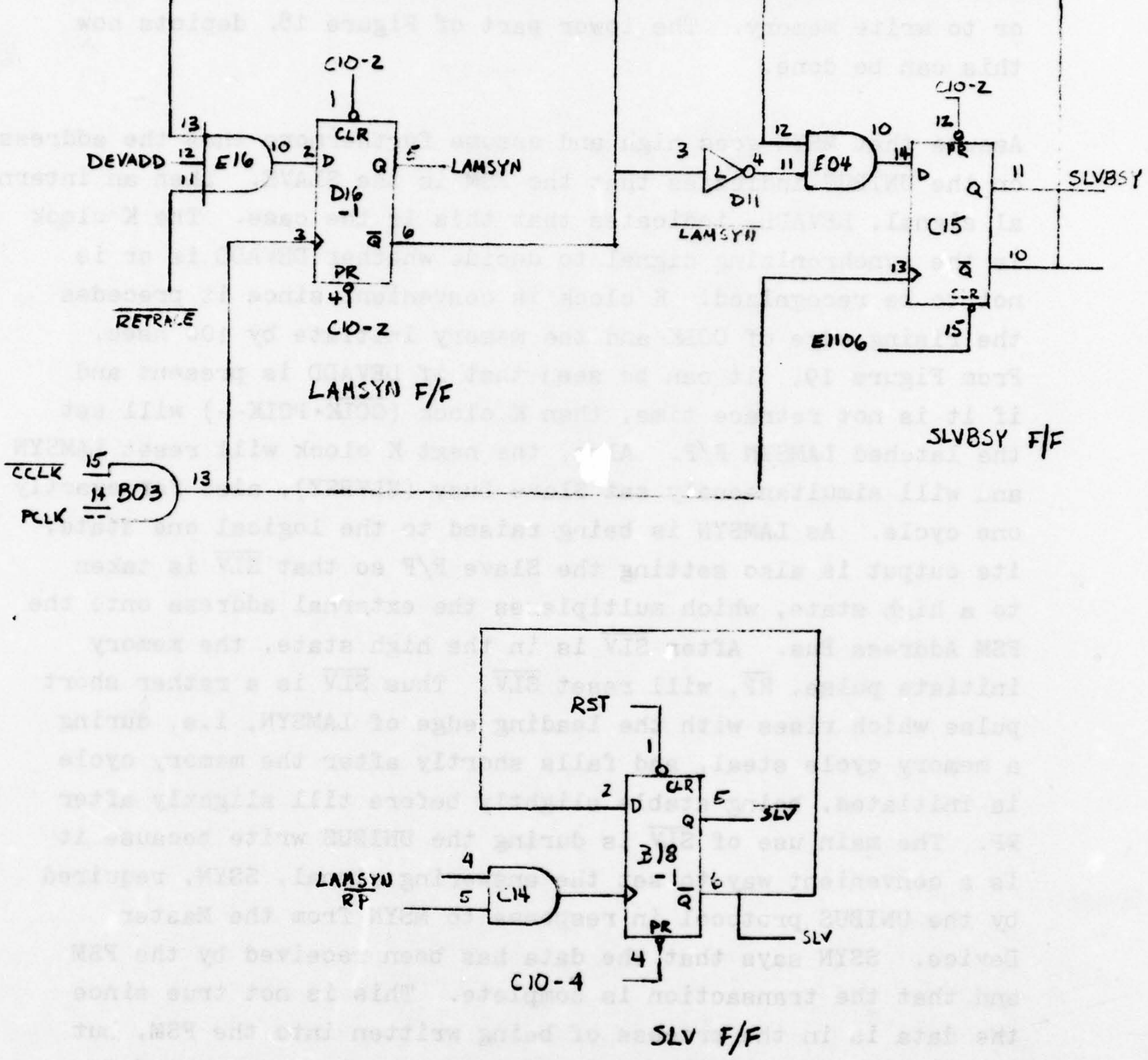


Figure 19. Cycle Steal Logic

When reading, the signal KCLK·SLVBSY is used to raise SSYN. Thus the K clock is doubly useful. First it provides the time reference for setting LAMSYN to steal a cycle for the UNIBUS; second, the next K clock (which is 300 ns after the memory initiate) provides a useful indication that Data Available has by now been generated by the Memory. Data Available occurs 250 ns after memory initiate.

The bottom half of Fig. 18. shows generation of SSYN depending on whether the stolen cycle was for reading or for writing and also shows when MSYN will drop -- following the rise of SSYN -- and how much time is left for it to be raised again for another cycle steal. When writing, MSYN can be dropped very early based upon SSYN being set by $\overline{\text{SLV}}$. When reading, MSYN is raised later based upon K·SLVBSY but still in adequate time to permit stealing every other cycle, if desired.

Three submodes can be distinguished when in the Slave Mode:

1. Active Line Time

This is the submode just discussed. In order for the UNIBUS to write or read, it must steal a cycle because normally in this submode the TV Monitor is being refreshed by pixel pairs being read out every 400 ns from the FSM. Fig. 18. and Fig. 19. have shown how this cycle steal can be accomplished.

2. Retrace Time

During horizontal retrace, which lasts 11.3 μs out of the 63.5 μs for the total TV line, there is no need to refresh the monitor. During this time CCLK is no longer permitted to trigger the memory. Instead, the UNIBUS is given sole access to the FSM. DEVADD·MB·$\overline{\text{SSYN}}$ is used to initiate the memory cycle, RP. DEVADD·MB is sufficient to carry out a memory cycle. The purpose of the additional term, $\overline{\text{SSYN}}$, is to prevent a double memory cycle in case MSYN is slow in being lowered when reading. Data Available at 250ns after memory initiate will set SSYN. Then MSYN should fall in 75 nsec. However, it is possible that it might still be high when the memory cycle is complete.

3. Fast Read

This is a special submode where there is no requirement to keep the monitor refreshed and where instead there is a need to read the FSM to the UNIBUS at the highest possible speed. To overcome the access time of the FSM, the word which is already in the Read Register when the UNIBUS sends an address and MSYN is placed on the UNIBUS and SSYN is sent. The first word sent thus has no meaning. However, as it is sent the address just received over the UNIBUS is used to address a word in FSM. By the time the Master device on the UNIBUS requests a second word, the first word is available and is sent. This is a very fast form of block reading based on overlapping the reads in the FSM and the UNIBUS transmission time.

Table 3. summarizes some of the important control signals which must be generated in the various modes. This table serves to specify the design of the actual logic. For example, the signal, TDA, a modified version of Data Available shown in Fig. 17. strobing the Read Register is implemented in Fig. 20. The signal RC6 refers to the CSR register set by the UNIBUS, specifically bit 6. When set, this bit specifies the Fast Read submode. Thus when in retrace or in Fast Read, DA strobes the Read Register; but when in Active Line, the K clock, as specified in Table 3, is the strobe.

It can be verified that the implementation of the clock to the SSYN F/F of Fig. 20. is exactly in agreement with the requirements of Table 3. Similiarly, Fig. 21. shows how RIN, the trigger to the RP pulse generation circuit in the Slave Mode, is implemented -- again in agreement with the requirements of Table 3. The trigger circuitry for the Master Mode is also shown for completeness.

At the top of Fig. 21. the Retrace F/F is shown. This F/F follows GUNBLK, which comes from the synchronization circuitry which times the TV Camera and Monitor. However, Retrace does not always immediately follow GUNBLK or GUNBLK but waits to com-

TABLE 3

SIGNALS REQUIRED TO BE GENERATED IN VARIOUS MODES

Mode	Generate $\overline{RP}$ to trigger memory	Clock the Read Register	Generate SSYN Signal for UNIBUS
1. Active Line			
1.1 Writing	CCLK		SLV
1.2 Reading	CCLK	K clock	KCLK·SLVBSY
2. Retrace			
2.1 Writing	DEVADD·$\overline{MB}$·$\overline{SSYN}$		RP
2.2 Reading	DEVADD·$\overline{MB}$·$\overline{SSYN}$	DA	DA
3. Fast Read			
3.1			
3.2 Reading	DEVADD·$\overline{MB}$·$\overline{SSYN}$	DA	RP

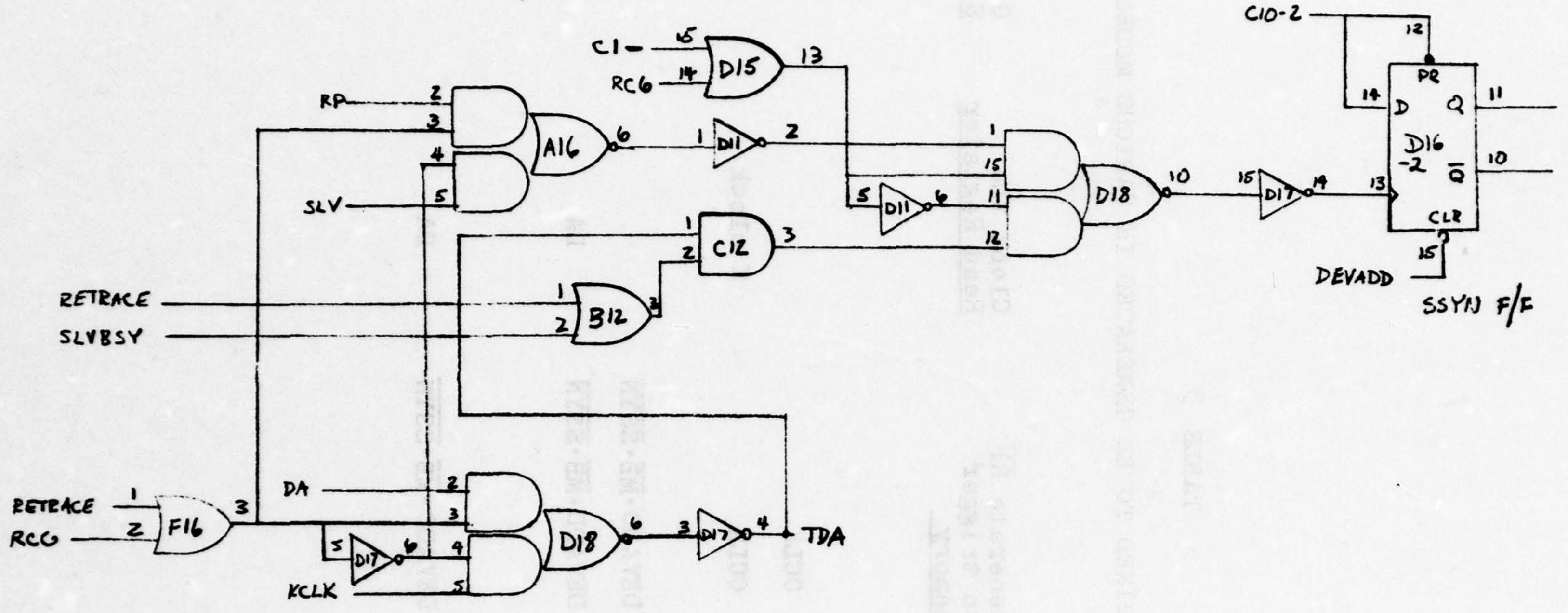


Figure 20. Read Clock and SSYN

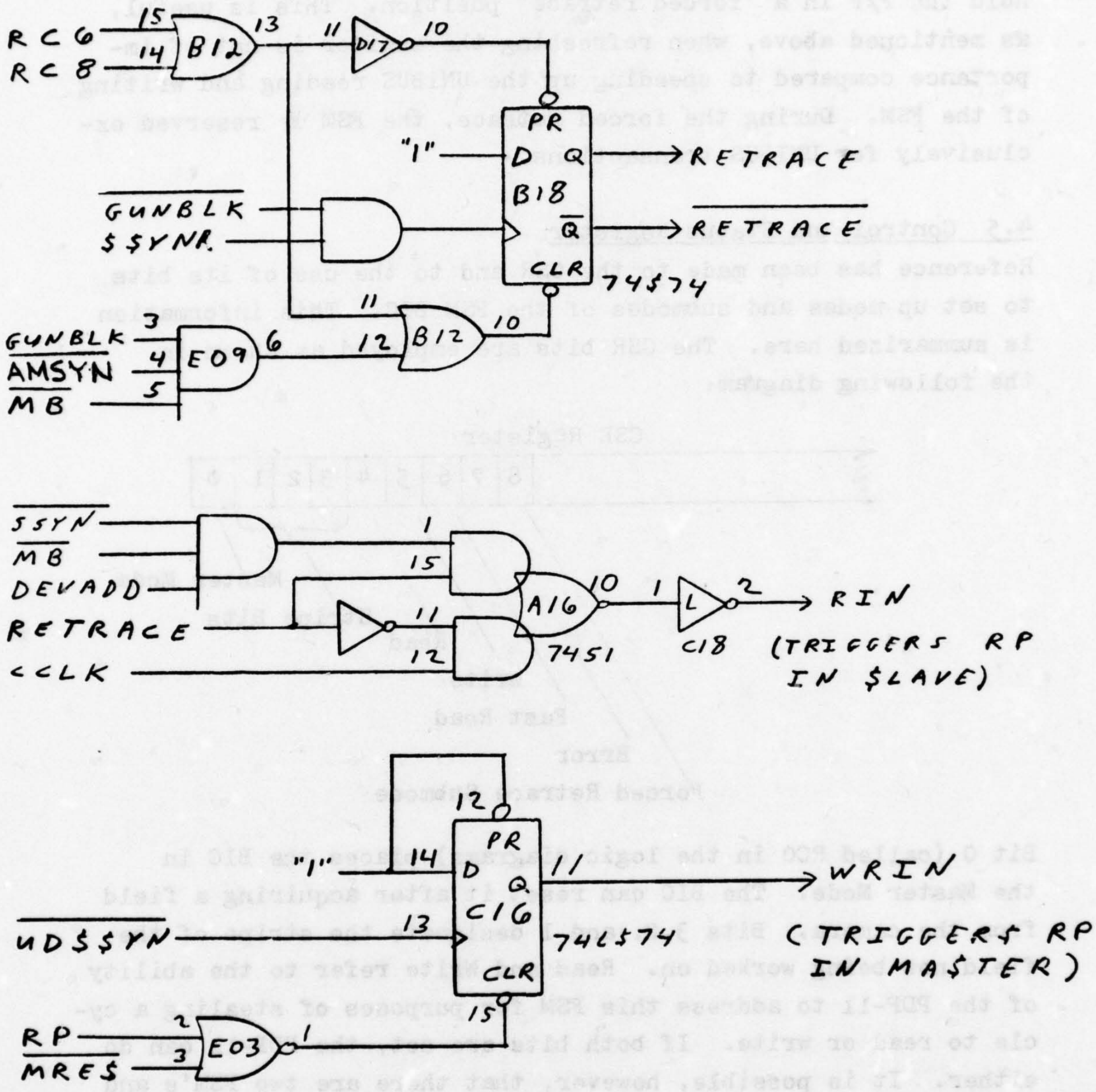
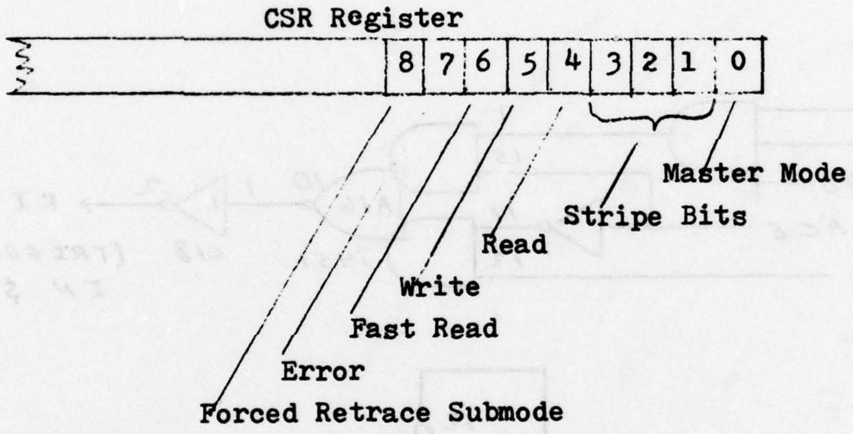


Figure 21. Retrace F/F and Memory Trigger

plete a transaction before changing from retrace to $\overline{\text{retrace}}$, or vice versa. Note that bit 8 of the CSR, designated RC8, can hold the F/F in a "forced retrace" position. This is useful, as mentioned above, when refreshing the monitor is not of importance compared to speeding up the UNIBUS reading and writing of the FSM. During the forced retrace, the FSM is reserved exclusively for UNIBUS transactions.

4.5 Control and Status Register

Reference has been made to the CSR and to the use of its bits to set up modes and submodes of the FSM BIC. This information is summarized here. The CSR bits are employed as shown in the following diagram:



Bit 0 (called RCO in the logic diagrams) places the BIC in the Master Mode. The BIC can reset it after acquiring a field from the camera. Bits 3,2, and 1 designate the stripe of the field now being worked on. Read and Write refer to the ability of the PDP-11 to address this FSM for purposes of stealing a cycle to read or write. If both bits are set, the PDP-11 can do either. It is possible, however, that there are two FSM's and the computer or microprocessor may be reading from one and writing to the other when they are both in the Slave Mode. As explained earlier, however, in lower order addressing with the PDP-11, to do a write to one FSM requires both Bits 4 and 5 to be set and the other FSM CSR Bits 4 and 5 to be clear. Fast Read and Forced Retrace have been discussed. Earlier it was mentioned that the Error Bit is set when in the Master Mode, no response is obtained within 300 µsec.

SECTION V

MICROPROCESSOR BUS INTERFACE CARD

In order for the microprocessor to carry out its DCT/DPCM computations, it must have its input FIFO filled and its output FIFO emptied. Thus the total burden of input/output is taken from the microprocessor and assigned to the Microprocessor Bus Interface Card (MP BIC). This card interfaces to the UNIBUS and not only handles all input/output for the microprocessor but also gives access to the internal registers of the microprocessor to the PDP-11 computer. Thus the computer can load the microprogram of the microprocessor, can force it to a starting address, can read the contents of various registers, can one-step the microprocessor, etc.

Because of the use of the 64-word First-In, First-Out memories at both input and output, the basic task of the MP BIC is to keep the input buffer filled and the output buffer emptied. As Bus Master, for example, the MP BIC can access 16 pixel pairs from the FSM during active line time. During horizontal retrace it can output 16 pixel pairs back to the FSM. Because of the short duration of a TV line (63.5 μsec) the MP BIC makes the highest level request to use the UNIBUS -- the Non-Processor Request (NPR). After a field (actually a stripe 32 pixels in width) has been processed, the BIC interrupts the PDP-11 computer at vertical retrace time. This permits the operator to halt operations if he desires to do so or to change the mode of operation.

5.1 System Modes

There are two configurations of the system and the BIC must be able to handle either. Configuration 1 has but a single microprocessor, while Configuration 2 uses two. These were discussed in an earlier section. Fig. 22. from that section lists pertinent information relative to inputs and outputs. This data can be summarized as in Fig. 23. It shows that the BIC must act as both Master and Slave and must be able to transfer blocks of 16 or 32 words to and from a variety of other units.

System Mode	Microprocessor Input	Input Control	Microprocessor Output	Output Control
1. Configuration 1 (Single processor)	16 pp (pixel pairs) from FSM	at horiz. sync but every other line	16 pp to the FSM	during horiz. or vertical retrace
2. Configuration 2				
2a. 1st micro-processor	16 pp from the TV camera BIC or from FSM	Following Gate Enable, at horiz sync, every line	32 12-bit coef-ficients to 2nd processor	after input, not horiz re-trace, or dur-ing vert retrac
2b. 2nd micro-processor	32 coefficients from 1st micro-processor	1st microproces-sor as Master will fill FIFO	16 pp to the FSM	during horiz. or vertical retrace

Figure 22. System Modes

__Acting as Bus Master__

__Input:__

16 words from camera BIC

16 words from FSM BIC

32 words from core memory

__Output:__

16 words to FSM

32 words to 2nd microprocessor

32 words to core memory

__Acting as Bus Slave__

__Input:__

32 words from 1st microprocessor

Figure 23. Microprocessor BIC I/O

Consider Mode 2a from Fig. 22. The BIC must, upon receiving the Gate Enable signal from the TV camera BIC, transfer 16 16-bit words from the TV BIC to the Input FIFO of the MP. This transfer must take place during the 52.2 μsec of active line time. This must be done for every line while the stripe is being processed. At the same time (actually during the same 52.2 μsec but after completion of inputting 16 words) the BIC must output 32 coefficients to the second microprocessor. Thus there are 48 UNIBUS transfers during active line time.

Consider Mode 2b of the BIC from Fig. 22. This is one corresponding to that of the second microprocessor, which is performing the inverse transforms. Because the first microprocessor acts as Master and fills the Input FIFO of the second microprocessor, the latter must take responsibility only for outputting 16 words to the FSM. It does this during the 11.3 usec of horizontal retrace.

It is clear that there is a delay from the camera to the 1st MP, from the 1st to the 2nd MP, and from the 2nd MP to the FSM. At vertical retrace time, there is still some data in the pipeline. The microprocessors have been designed so that they continue to compute as long as there is data in the Input FIFO. When there is no data, they stop computing until there is data available. Thus, although vertical retrace has occurred, the MP's continue to compute until all data is passed through the system. This might be referred to as a "fall-through" method of processing the data.

Upon completion of processing a stripe, the microprocessors interrupt the computer so that it might decide what is to be done next. Typically the PDP-11 will simply update the stripe information so that the processing will move on to the next stripe to the right of the one just processed. The PDP-11 can, however, suspend the operations or call for a different type of processing.

5.2 Functional Description

The MICROPROCESSOR BUS INTERFACE CARD (MP BIC) block diagram is shown in Figure 24. The five major areas of the MP BIC are the UNIBUS Driver/Receivers, the Microprocessor Driver/Receivers, the Control Logic, the UNIBUS Addressing Logic, and the Microprocessor Addressing Logic.

5.2.1 UNIBUS Driver/Receiver

The UNIBUS Driver/Receivers provide the communication paths to and from the PDP-11 UNIBUS. Implemented with AMP26S10's quad bus transceivers with open collector outputs, they allow the MP BIC to send and receive the UNIBUS address, data, and control signals.

The UNIBUS has 18 address signal lines (A17-A00) which allow addressing up to 256K bytes. However, since the MP BIC performs only full word (2 bytes) transfers, the least significant address bit (A00) is not implemented.

A UNIBUS data word is 16 bits in length. In addition to handling normal data words to and from the UNIBUS these driver/receivers provide the MP BIC with a path for an interrupt vector which is sent to the CPU. This interrupt vector is 9 bits wide (D08-00) and identifies the source of interrupt. The MP BIC is assigned interrupt vectors 170_8 and 270_8.

Also transfered via these driver/receivers is a status bit generated by the MP BIC which is supplied with the microprocessor status. This status bit indicates when the output FIFO of the microprocessor I/O board is empty.

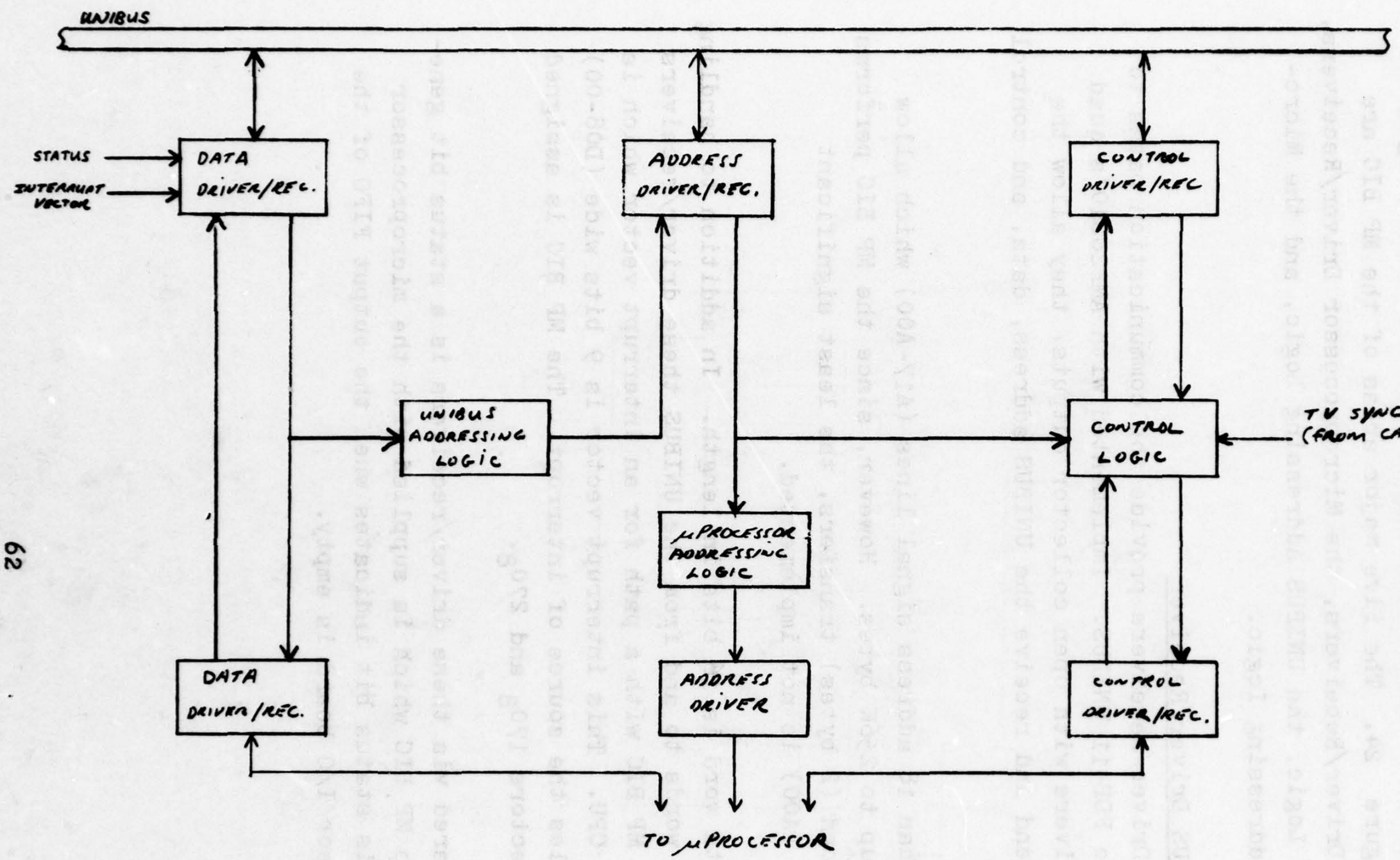


Figure 24. Microprocessor Bus Interface Card (MP BIC) Block Diagram

5.2.2 Microprocessor Driver/Receivers

The microprocessor driver/receiver provide the communication paths to and from the microprocessor (MP). There are four address signals which select registers and functions within the MP. This allows for 32 individual addresses; 16 for sending data to the MP and 16 for fetching data from the MP. These addresses are identified in Table 4. Refer to Section 6 for a detailed description of these registers.

The MP data signals consist of 16 bidirectional lines used to exchange data between the MP BIC and the MP. In most cases this is the same data which is received from or sent to the UNIBUS. There are five control signals which are sent from the MP BIC to the MP and two signals which are sent back from the MP. The five signals sent by the MP BIC are:

1. INIT (Initiate) which is sent over the UNIBUS and causes a master reset of the MP.
2. SIIF (Shift In Input FIFO) which causes the input FIFO of the MP to shift in data from the MP BIC.
3. SOOF (Shift Out Output FIFO) which causes the output F FIFO of the MP to shift out data to the MP BIC.
4. DATAMP (Data to the MP) which controls the direction of the MP data signals.
5. SSTB (Slave Strobe) which strobes data into the MP when the MP BIC is receiving data in the slave mode.

The two signals sent by the MP consist of IFIR (Input FIFO input ready) and OFOR (Output FIFO output ready) which inform the MP BIC when data can be sent to or received from the MP, respectively.

5.2.3 Control Logic

The control logic of the MP BIC supervises the transmission of data between the UNIBUS and the MP. A block diagram is shown in Figure 25. This control logic can be divided into two major groups: The slave mode logic and the master mode logic.

TABLE 4

MICROPROCESSOR ADDRESSABLE REGISTERS

ADDRESS				DESCRIPTION	
MSB			LSB	DATA TO MP (WRITE)	DATA FROM MP (READ)
0	0	0	0	RESET △1	DATA REGISTER
0	0	0	1	HALT △1	
0	0	1	0		Scratch Pad Register
0	0	1	1		
0	1	0	0	Destination Address Reg. △2	
0	1	0	1	Source Address Reg. △2	
0	1	1	0		
0	1	1	1	Input FIFO	Output FIFO
1	0	0	0	START △1	
1	0	0	1	Control Status Reg.	Control Status Reg. △3
1	0	1	0		
1	0	1	1	Breakpoint Reg.	
1	1	0	0	Address Counter	Address Counter
1	1	0	1	Control Store 1 (Bits 15-0)	Control Store 1 (Bits 15-0)
1	1	1	0	Control Store 2 (Bits 31-16)	Control Store 2 (Bits 31-16)
1	1	1	1	Control Store 3 (Bits 47-32)	Control Store 3 (Bits 47-32)

△1 Addressing of these locations executes the function specified. They are not registers.

△2 Located on the MP BIC.

△3 Some Status Bits are located on the MP BIC.

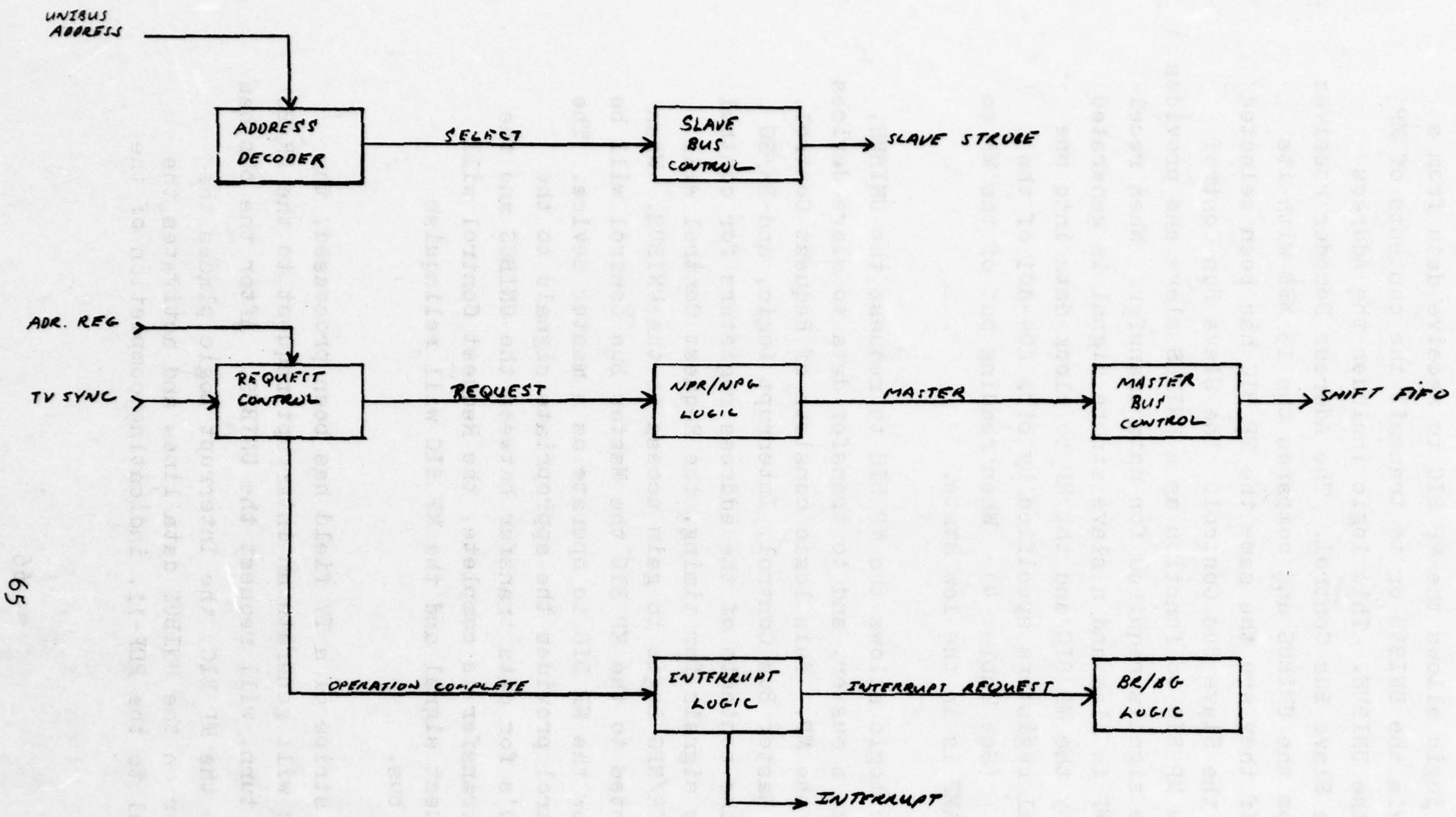


Figure 25. MP BIC Control Logic Block Diagram

The slave mode logic allows the MP BIC to receive data from a master device via the UNIBUS or to transmit the contents of MP register over the UNIBUS. This logic includes the Address Decoder and the Slave Bus Control. The Address Decoder receives the address from the UNIBUS and compares the 13 MSB with its own address. If they are the same the MP BIC has been selected and so enables the Slave Bus Control. The Slave Bus Control then causes the MP BIC to function as a UNIBUS slave and provides the appropriate signals required for data transfer. When receiving data DATAMP is high and a slave strobe signal is generated which is used by the MP BIC and the MP to clock data into one of 16 individual registers specified by bits A04-A01 of the UNIBUS address. (See Table 4) When reading out of the MP to the UNIBUS DATAMP is in the low state.

The master mode logic allows the MP BIC to request the UNIBUS, to control it as a master, and to transfer data to slave devices as required by the MP. This logic consists of Request Control, NPR/NPG Logic, Master Bus Control, Interrupt Logic, and BR/BG Logic. Using the contents of the address registers for control and the TV sync signals for timing, the Request Control will activate the NPR/NPG Logic to gain access to the UNIBUS. When the bus is granted to the MP BIC the Master Bus Control will be enabled to allow the MP BIC to operate as a master device. The Master Bus Control provides the appropriate signals to the UNIBUS and FIFO's for data transfer between the UNIBUS and the MP. When the transfer is complete, the Request Control will remove the request signal and the MP BIC will relinquish control of the bus.

When an entire stripe of a TV field has been processed, the Interrupt logic will generate an interrupt request to the BR/BG Logic which in turn, will request the UNIBUS. After the bus has been granted to the MP BIC, the Interrupt Logic places the interrupt vector on the UNIBUS data lines and activates the interrupt signal to the PDP-11, indicating completion of the processing.

5.2.4 UNIBUS Addressing Logic

In the master mode the MP BIC provides source addresses for UNIBUS Data-In operations (fetch) and destination addresses for Data-Out operations (stores). These addresses are generated by the UNIBUS Addressing Logic which includes Source and Destination Address Registers, Source and Destination Address Counters, Address Mux, and Corner Turning Mux. The block diagram is shown in Figure 26.

Data contained in the Source and Destination Address Registers (SAR and DAR, respectively) is used by the MP BIC for both UNIBUS Addressing and Control Logic operation. These registers are addressed and loaded via the UNIBUS and occupy two of the 16 write addresses assigned to the MP. (See Table 4)

The format for the 16 bit SAR is shown in Figure 27. Bits 15-11 are used for control while bits 7-1 provide addresses.

FSM:	When set (true) this bit causes the Control Logic to fetch data from the Frame Store Memory.
CORE:	When set this bit causes the Control Logic to fetch data from core memory.
CAM:	When set this bit causes the Control Logic to fetch data from the camera.
ALT:	When set this bit causes the Control Logic to fetch data during alternate TV lines. This is typically used only in Configuration 1.
CT:	When set this bit selects the Corner Turning Mux so that the source address bits are arranged for Corner Turning Addressing of the core memory to fetch "vertically".
ADDRESS:	These bits are used to form part of the UNIBUS address when fetching.

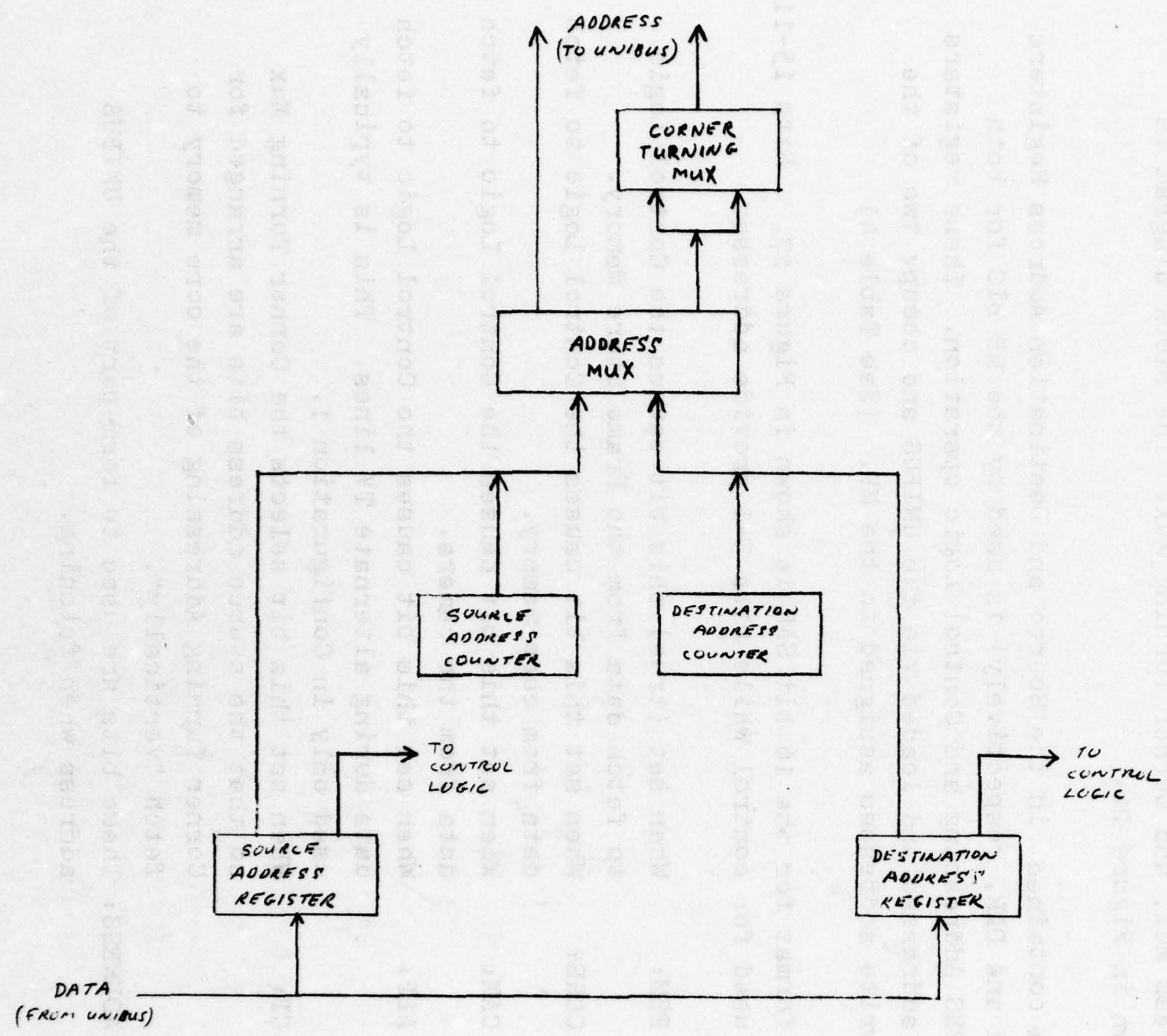


Figure 26. UNIBUS Addressing Logic Block Diagram

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FSM	CORE	CAM	ALT	CT	NOT USED			ADDRESS							NOT USED

Figure 27. SAR Format

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FSM	CORE	NOT USED	MP	CT	NOT USED			ADDRESS							NOT USED

Figure 28. DAR Format

The format for the 16 bit DAR is shown in Figure 28. Bits 15, 14, 12 and 11 are used for control and bits 7-1 for addressing.

FSM:	When set this bit causes the Control Logic to store data to the Frame Store Memory.
CORE:	When set this bit causes the Control Logic to store data to the core memory.
MP:	When set this bit causes the Control Logic to store data to the second microprocessor in a system with two microprocessors.
CT:	When set this bit selects the Corner Turning Mux so that the destination address bits are arranged for core memory Corner Turning in two-dimensional processing.
ADDRESS:	These bits are used to form part of the UNIBUS ad address when storing.

The use of the SAR/DAR words is illustrated by means of Fig. 29. The first line, for example, shows that the source of data will be FSM1 and that, depending upon whether there is one microprocessor or two in the system (bit 12 of the SAR) 16 pixel pairs will be accessed every other line time or every line.

The Source and Destination Counters (SAC and DAC, respectively) are 13-bit counters which form the least significant portion of the UNIBUS address when transfering data to or from the Frame Store Memory or core memory.

The Address Mux is a 17 bit wide multiplexer which selects the proper bits for the UNIBUS address. The bits and the conditions which select them are shown in Table 5.

The Corner Turning Mux is a 10 bit wide multiplexer which controls the 10 least significant output bits of the Address Mux (AM10-AM01) in order to provide the corner turning feature. This is selected by bit 11 of the SAR when fetching and by bit 11 of the DAR when storing.

SOURCE	SAR BITS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2		
FSM1	1	0	0	B	0	X	X	X	1	1	1	1	0	X	X	X
FSM2	1	0	0	B	0	X	X	X	1	1	1	0	1	X	X	X
FSM3	1	0	0	B	0	X	X	X	1	1	1	0	0	X	X	X
CORE	0	1	0	0	A	X	X	X	0	0	0	1	X	X	X	X
CAMERA	0	0	1	0	0	X	X	X	0	0	1	0	0	1	0	X
NONE	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X

DESTINATION	DAR BITS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSM1	1	0	X	0	0	X	X	X	1	1	1	1	0	X	X	X
FSM2	1	0	X	0	0	X	X	X	1	1	1	0	1	X	X	X
FSM3	1	0	X	0	0	X	X	X	1	1	1	0	0	X	X	X
CORE	0	1	X	0	A	X	X	X	0	0	0	1	X	X	X	X
MP1	0	0	X	1	0	X	X	X	0	0	0	0	0	1	1	X
MP2	0	0	X	1	0	X	X	X	0	0	0	1	0	1	1	X

A: 0 = no corner turning
1 = corner turning

B: 0= system modes 2 or 3
1= system mode 1

Fig. 29. Specific SAR/DAR Words

TABLE 5

ADDRESS MUX OUTPUT CONFIGURATION

		SOURCE (FETCH)			DESTINATION (STORE)		
		FSM	CORE	CAM	FSM	CORE	MP
	BIT	SAR 15	SAR 14	SAR 13	DAR 15	DAR 14	DAR 12
(MS)	17	SAR 07	SAR 07	1	DAR 07	DAR 07	1
	16	06	06	1	06	06	1
	15	05	05	1	05	05	1
	14	04	SAR 04	1	04	DAR 04	1
	13	SAR 03	SAC 12	1	DAR 03	DAC 12	1
ADDRESS	12	SAC 11	11	0	DAC 11	11	0
MUX	11	10	10	1	10	10	1
OUTPUT	10	09	09	0	09	09	0
	09	08	08	0	08	08	0
	08	07	07	SAR 07	07	07	DAR 07
	07	06	06	06	06	06	06
	06	05	05	05	05	05	05
	05	04	04	04	04	04	04
	04	03	03	03	03	03	03
	03	02	02	02	02	02	02
	02	01	01	SAR 01	01	01	DAR 01
(LS)	01	SAC 00	SAC 00	1	DAC 00	DAC 00	1

When the corner turning bit is not set (false) the output of the Address Mux is passed, unaltered, to the UNIBUS. When the bit is set the five least significant bits of the Address Mux (AM05-01) are interchanged with the next five bits (AM10-AM06).

5.2.5 Microprocessor Addressing Logic

The Microprocessor Addressing Logic is a 4 bit wide multiplexer which selects the address bits that go to the MP. When the MP BIC is in the master mode the Mux always addresses the MP FIFO's (code 0111 in Table 4). If the MP BIC is not in the master mode, the UNIBUS address bits A04-A01 are passed unaltered to the MP.

The significance of the above can be seen from an examination of Table 4, which relates the least significant four bits of the address to the appropriate register within the Microprocessor or to the function which is triggered when this address is decoded by the MP BIC. When the MP BIC is in the master mode, it must itself control the MP. It does its fetching and storing of data from or to the Output FIFO or Input FIFO. Both are given the same address, 0111, but there is no confusion because the direction of data flow (Read or Write) makes clear which one is to be used.

In the slave mode, the BIC simply passes through to the MP the four least significant bits from the UNIBUS address. When the computer is addressing the MP, for example, it may be writing into or reading from the Control & Status Register (code 1001); or it can write to the Destination Address Register, Source Address Register, Breakpoint Register, Address Counter, Control Store word 1, 2, or 3, etc. Similarly the computer can read most of these registers.

When the computer "writes" to address 0000 or 0001, it really is not writing data to a register. Instead, when the BIC decodes 0000 during a write, it carries out a reset. When it decodes a 0001, it halts the MP.

5.3 Operation

The MP BIC can operate as either a UNIBUS Master or Slave. A single microprocessor's Bus Interface Card acts as a Master both for data input and data output. If there is also a second MP (Configuration 2) it acts as a Slave for receiving data from the first MP and as a Master for data output to the FSM. When it is not acting as Bus Master, a MP BIC can be addressed as a Slave device and be given commands by the PDP-11 computer. At this time its registers can be loaded or read out, and its operations controlled with respect to start, stop, etc.

5.3.1 Slave Mode

In the Slave mode the MP registers can be either written into or read from by the CPU. The receipt of MSYN and $C_1=1$ (which indicates Data output from the Master) causes the BIC to strobe UNIBUS data into the register specified by UNIBUS address bits A04-A01 (see Table 4). The strobe, SSTB, goes true after MSYN is received and remains true until the BIC sends SSYN to the bus and causes MSYN to drop.

If MSYN is received and $C_1=0$ (Data In to the Master), the BIC enables the contents of the register specified by address bits A04-01 onto the bus and sends SSYN to the requesting device. This data remains on the bus until MSYN is dropped.

As a master device the PDP-11 Computer can read the output FIFO of the microprocessor or can write into the input FIFO. In a two microprocessor system the first MP can send data to the second MP's input FIFO. Hence the MP BIC must be able to respond as a Slave to a Master device's writing to FIFO or reading from FIFO. It does so by generating a signal to "shift in" to the Input FIFO or "shift out" of the Output FIFO, as appropriate. These "shift" signals are generated by the MP BIC upon receipt of MSYN from the Master device.

5.3.2 Master Mode

The MP BIC acts as a Master device for data input and output based upon control information passed to it by the computer. The PDP-11 can control the operation of the BIC by loading the Source Address Register (SAR) for data input and the Destination Address Register (DAR) for data output (refer to Figures 27 to 29). The conditions which determine when the BIC will perform as a Master include TV sync signals, UNIBUS accessibility, and data availability.

In the Master mode, the MP BIC has three states; fetch, store, and wait.

FETCH:	In this state the MP BIC requests the bus so that it may fetch data from the source device specified in the SAR. When control is granted, the BIC fetches either 16 or 32 words and shifts it into the input FIFO of the MP.
STORE:	In this state the MP BIC requests the bus so that it may store data to the destination device specified in the DAR. When control is granted, the BIC shifts data out of the MP output FIFO as required and stores the data via the UNIBUS.
WAIT:	If the MP BIC is not required to fetch data (fetch state) or to store data (store state) then no UNIBUS requests are made and the MP BIC remains in the wait state.

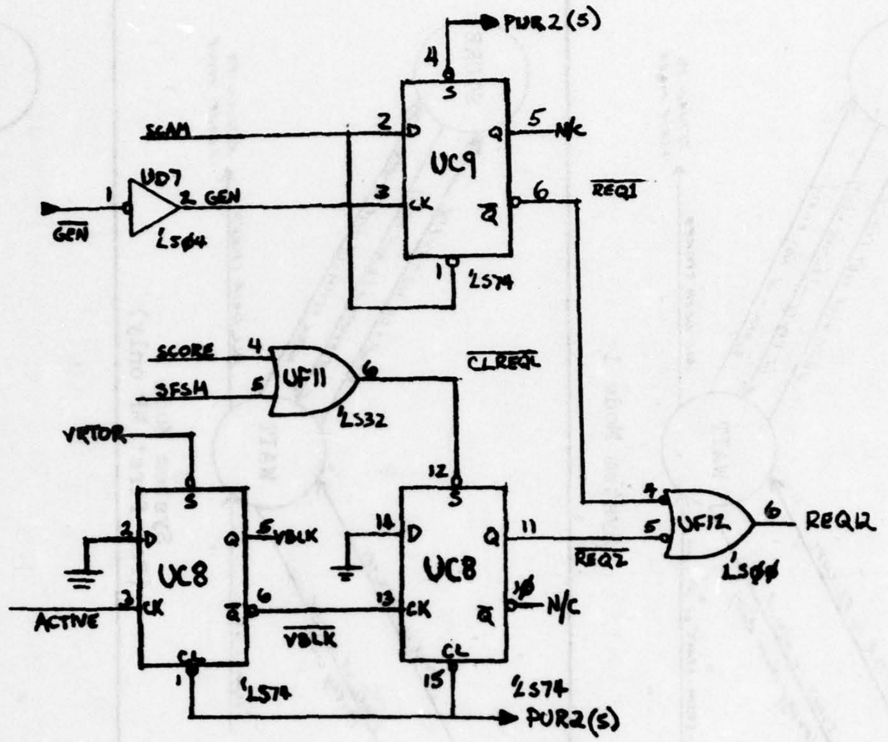
The MP BIC is designed to operate in three system modes. These modes (one MP or two MP's performing a 1-Dimensional DCT followed by DPCM or one MP doing a 2-Dimensional DCT) were discussed in Section 5.1. The following information describes the operation of the MP BIC in each of these modes.

System Mode 1

In this mode data is fetched from the FSM, processed by the MP, (which performs both forward and inverse transforms) and finally stored once again in the FSM. Bit 15 "SFSM", of the SAR is true indicating that the source of data is the FSM. This condition enables the Fetch Request Sync Logic (Figure 30) by forcing $\overline{\text{CLREQ2}}$ high. This allows the first active TV line (ACTIVE) during vertical drive (VRTDR) to force $\overline{\text{REQ2}}$ low. This synchronizes the start of fetching to the top of a field and causes the MP BIC to enter the master mode as shown in Figure 31.

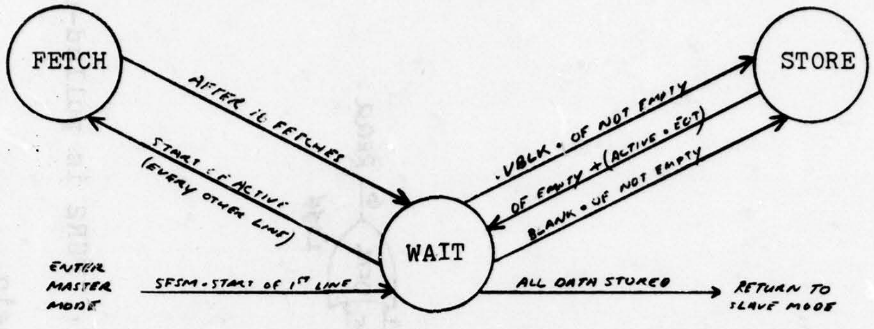
In addition, at the end of the first active line time ALT is forced true by the leading edge of BLANK (refer to Figure 32). REQ12, from sync logic, is anded with bit 12 (SALT) of the SAR and ALT which forces REQA high allowing SREQA to go true at the start of the next active line (ACTIVE) putting the BIC in the fetch state. This causes the MP BIC to request the bus via its NPR/NPG logic which was discussed in Section 5.2 (refer to Figure 25). When control is granted the BIC becomes UNIBUS Master and places address and control on the bus and rasies MSYN which causes EOT of the Master Bus Control to go false. When SSYN is received the data is shifted into the input FIFO and MSYN is dropped causing EOT to go true which indicates the end of transfer. EOT, in turn, increments the Source Address Counter (SAC). (SAC is a 12 bit counter which is set to zero whenever SAR is loaded and is used to supply the least significant address bits when addressing FSM.)

After the 15th fetch, SAC raises SCT16 (Figure 32) which is anded with $\overline{\text{SCORE}}$ (fetch from core, not). Therefore, when $\overline{\text{EOT}}$ goes low at the beginning of the 16th fetch it forces $\overline{\text{STREM}}$ low. At the end of the 16th fetch, EOT cause $\overline{\text{CLREQA}}$ to go low which forces SREQA low. This removes the request from the NPR/NPG logic and the BIC relinguishes control of the bus and goes to the wait state.

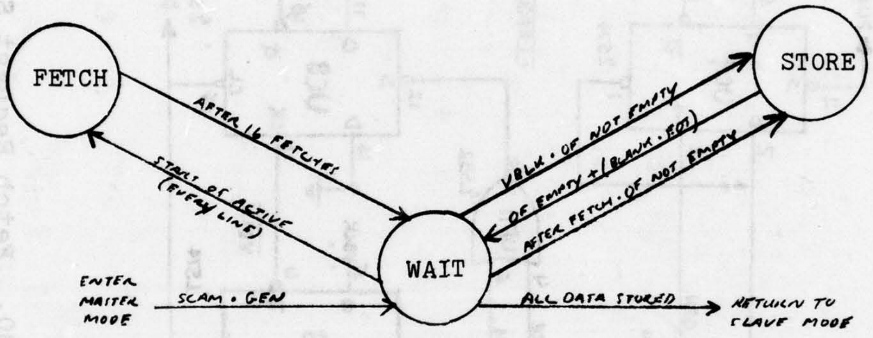


NOTE: PUR2 is pulled-up to VCC

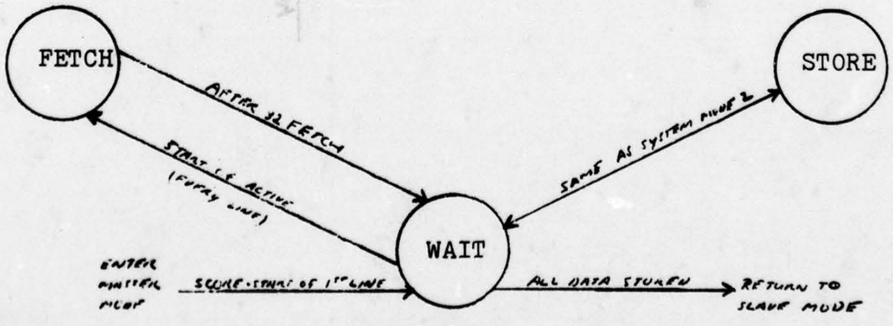
Figure 30. Fetch Request Sync Logic



System Mode 1

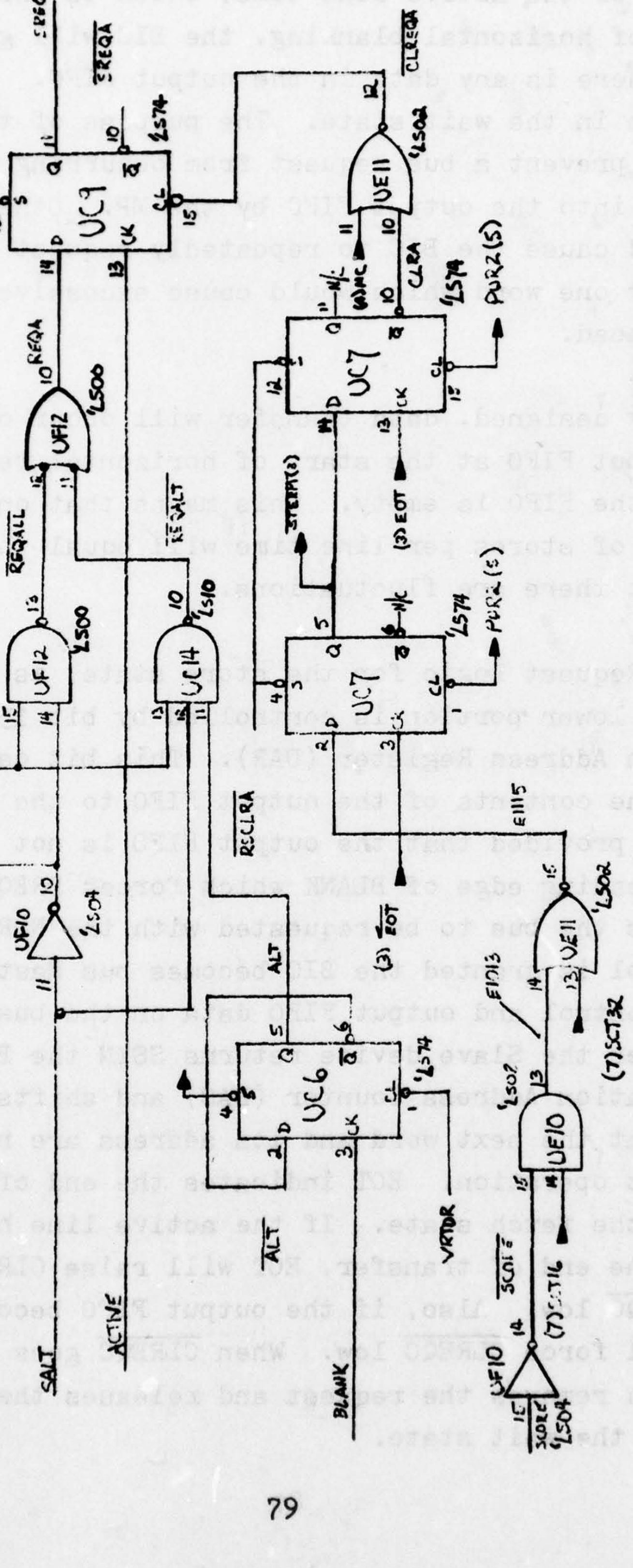


System Mode 2
(For first MP only)



System Mode 3
(For Core Memory only)

Figure 31. State Diagrams



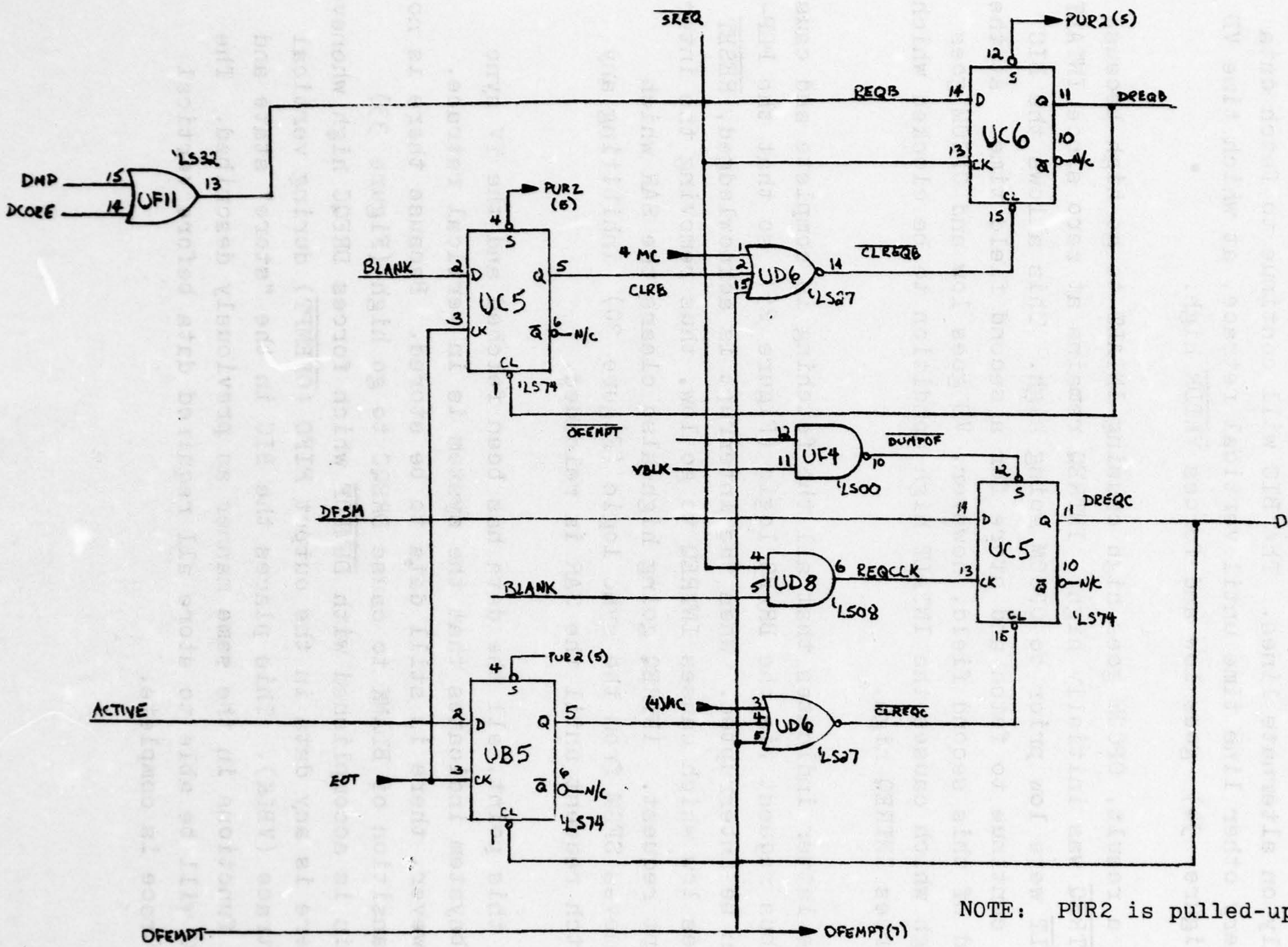
NOTE: PUR2 is pulled-up to VCC

Figure 32. Fetch Request Logic

The MP BIC is in the fetch state for approximately 10 μs and then in the wait state for the remainder of the active line time. At the end of the active line time, which is the same as the beginning of horizontal blanking, the BIC will go to the store state if there is any data in the output FIFO. If not, the BIC will remain in the wait state. The purpose of this design feature is to prevent a bus request from occurring each time a word is shifted into the output FIFO by the MP. Otherwise, the latter event would cause the BIC to repeatedly request the bus in order to transfer one word which would cause excessive bus request and grant overhead.

As actually designed, data transfer will occur only if data is in the output FIFO at the start of horizontal retrace and will stop when the FIFO is empty. This means that on the average the number of stores per line time will equal the number of fetches but there are fluctuations.

The Store Request Logic for the store state, is shown in Figure 33. The lower portion is controlled by bit 15 (DFSM) of the Destination Address Register (DAR). This bit causes the MP BIC to store the contents of the output FIFO to the FSM. This is initiated, provided that the output FIFO is not empty (OFEMPT), with the leading edge of BLANK which forces DREQC high. This also causes the bus to be requested with the NPR/NPG logic. When control is granted the BIC becomes bus Master and places address, control and output FIFO data on the bus and raises MSYN. After the Slave device returns SSYN the BIC increments the Destination Address Counter (DAC) and shifts the output FIFO so that the next word and its address are ready for the next output operation. EOT indicates the end of transfer, as it did in the fetch state. If the active line has occurred prior to the end of transfer, EOT will raise CLRC which will force $\overline{\text{CLREQC}}$ low. Also, if the output FIFO becomes empty, OFEMPT will force $\overline{\text{CLREQC}}$ low. When $\overline{\text{CLREQC}}$ goes low it drops DREQC which removes the request and releases the bus, returning the BIC to the wait state.



NOTE: PUR2 is pulled-up to VCC

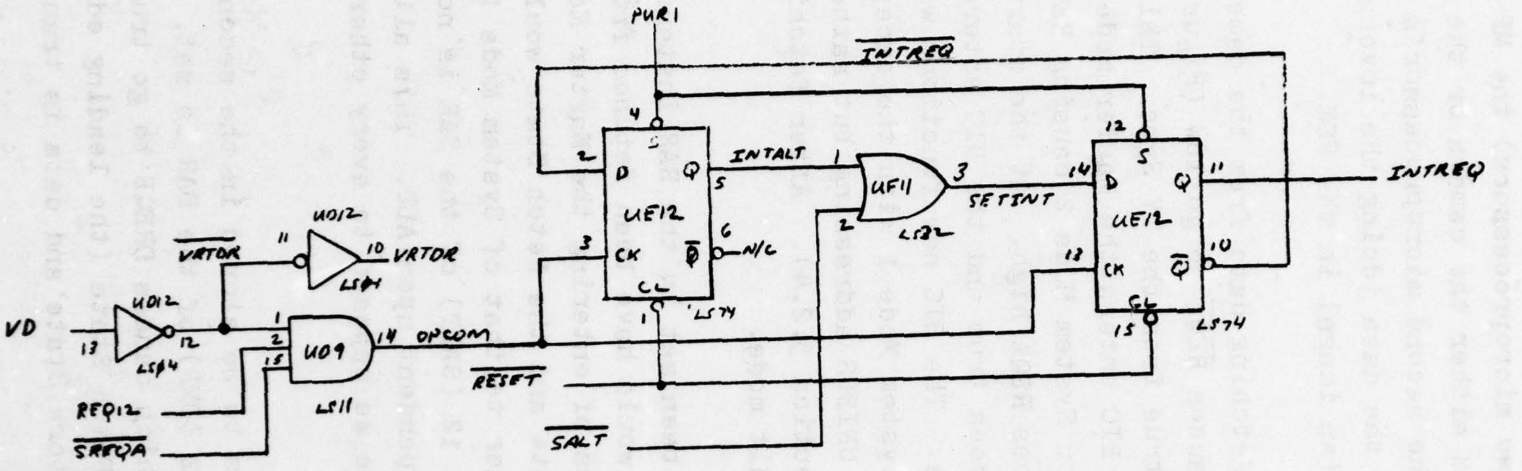
Figure 33. Store Request Logic

The leading edge of BLANK, which was discussed above, also forces ALT low which inhibits SREQA from going true. Each sucessive BLANK will toggle ALT and permit the BIC to enter the fetch state only on alternate lines. The BIC will continue to fetch data every other line time until vertical retrace, at which time VD (Figure 34) goes low and forces $\overline{\text{VRTDR}}$ high.

As a result, OPCOM goes high causing INTALT to go high because $\overline{\text{INTREQ}}$ was initially high. INTREQ remains at zero since INTALT and $\overline{\text{SALT}}$ were low prior to OPCOM going high. This allows the BIC to continue to fetch and store for a second field time. At the end of this second field, however, VD goes low and OPCOM goes high which causes the INTALT high condition to be clocked which forces INTREQ high.

The latter indicates that all the fetching is complete and causes a bus request via the BR/BG logic (Figure 25) so that the PDP-11 can be interrupted. When the interrupt is acknowledged, $\overline{\text{RESET}}$ goes low which causes INTREQ to go low, thus removing the interrupt request. INTREQ going high also clears the SAR which removes SFSM from the sync logic (Figure 30) inhibiting any fetch request until the SAR is reloaded.

At this point, all the data has been fetched and the TV sync subsystem indicates that the system is in vertical retrace. However, there is still data to be stored. Because there is no transition of BLANK to cause DREQC to go high (Figure 33) this is accomplished with $\overline{\text{DUMPOF}}$ which forces DREQC high whenever there is any data in the output FIFO ($\overline{\text{OFEMPT}}$) during vertical retrace (VBLK). This places the BIC in the "store" state and it functions in the same manner as previously described. The BIC will be able to store all required data before vertical retrace is complete.



NOTE: PUR1 is pulled-up to VCC

Figure 34. Termination Logic

System Mode 2

In System Mode 2 (two microprocessors) the MP BIC of the first MP fetches data from either the camera or the FSM, processes it, and stores it in the second microprocessor's input FIFO. The second MP processes the data (doing the inverse transform) and stores the results (an image) in the FSM.

If the first MP is fetching data from the camera then bit 13 (SCAM) of the SAR causes $\overline{\text{REQ1}}$ to go low (Figure 30) at the time the first GEN goes true from the TV Sync. This causes REQ12 to go high, and the MP BIC enters the Master mode. Bit 12 (SALT) of the SAR is low for System Mode 2 causing REQ12 to be anded with $\overline{\text{SALT}}$ which forces REQA high. At the start of the next active line, SREQA goes true and the BIC enters the fetch state and requests the bus. The BIC now functions within this state just as it did for System Mode 1 with the exception that the SAC is not used for the UNIBUS addressing but rather a fixed address is used (refer to Section 5.2.4). After fetching 16 pixel pairs it returns to the wait mode.

If bit 15 (SFSM) had been set in the SAR instead of bit 13 (SCAM) then 16 pixel pairs would have been fetched from the FSM. In this case the process of entering the Master Mode and switching between the Wait State and the Fetch State would be carried out in an identical manner to that of System Mode 1. But note that in System Mode 2 bit 12 (SALT) of the SAR is not set and, therefore, REQA is not dependent upon ALT. This allows fetching to occur every line time as opposed to every other line time for System Mode 1.

The processed data is to be stored in the second MP's input FIFO and therefore, bit 12 (DMP) of the DAR is set. This forces REQB true (Figure 33) which causes DREQB to go true as soon as the MP returns from the Fetch State (the leading edge of $\overline{\text{SREQ}}$).
The BIC enters the Store State and data is transfered from the

first MP's output FIFO to the second MP's input FIFO. These store operations via the UNIBUS are the same as for System Mode 1 except that the DAC does not provide the addressing. Instead, a fixed address is used (refer to Section 5.2.4). The MP BIC returns to the Wait State at the end of a transfer (EOT) provided that horizontal or vertical retrace has occurred. This causes CLRB to go high (refer to Figure 33). The BIC also returns to the Wait State if the output FIFO becomes empty (OFEMPT). Either of these conditions cause $\overline{\text{CLREQB}}$ to go low, forcing DREQB low, thus removing the request for the bus.

These fetch and store operations continue until vertical retrace occurs (VD goes low) forcing $\overline{\text{VRTDR}}$ and OPCOM high (Figure 34) Unlike System Mode 1, $\overline{\text{SALT}}$ is true and the leading edge of the first OPCOM forces INTREQ to go true, thus terminating the operation at the end of each field. The interrupt process, clearing of SAR, and storing the remainder of the data, is done just as for System Mode 1.

The second MP receives the data as a Slave device and hence does not have to enter a Fetch State. This is controlled by zeroing bit 15 (SFSM), bit 14 (SCORE), and bit 13 (SCAM) of the SAR, thus inhibiting REQ12 of the Fetch Request Sync Logic (Figure 30) from ever going true. Outputting of data from the second MP to the FSM is handled in the same manner as that described for System Mode 1.

SECTION VI

MODEL 1240 MICROPROCESSOR

Built up from three Am 2901 4-bit slice Schottky TTL microprocessor chips, the Model 1240 is a very high-performance processor which is ideally suited for signal processing applications. A block diagram of the Am 2901 chip is shown in Figure 35 and that of the Model 1240 in Figure 36. From Figure 35 it can be seen that the Am 2901's ALU accepts two inputs, both of which can come from a dual port RAM in which there are 16 general registers. Each microinstruction combines arithmetically or logically two operands and places the result, unshifted or shifted right or left, back in the second general register. This result can also be put on the data output bus. Instead of taking both operands from general registers, the Am 2901 can accept one input from the Direct Data bus. System cycle times as short as 150 ns are possible with the Am 2901.

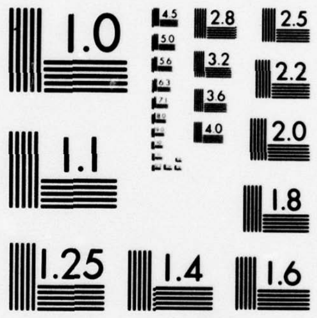
In the Model 1240, pipeline architecture implemented with Schottky TTL circuit provides very fast processing. The Am 2901 elements and other essential subsystems, including scratch pad memory and asynchronous multiplier, have been carefully organized to produce a system structure that can be programmed using a high degree of pipelining. Propagation delays through the various system elements are matched using data latches where necessary and numerous data paths are available so that parallel transfers are possible.

To realize the full potential of this pipeline architecture programs are typically coded entirely in microcode using straight line programming. Using this method a maximum number of parallel operations can occur and the inefficiencies attributable to a hierarchy of code, and resultant program branches, are eliminated.

The microprocessor (MP) is designed so that it can be conveniently interfaced to a host computer as a peripheral device. A special Bus Interface Card (BIC) provides the proper format and protocol required by the host computer.

AD-A072 917   DATA/WARE DEVELOPMENT INC SAN DIEGO CALIF   F/G 9/2
ADVANCED DIGITAL TV SYSTEM.(U)
FEB 79 P J ERDELSKY, R V KEELE, G G MURRAY   F33615-77-C-1198
UNCLASSIFIED   AFAL-TR-79-1006   NL

2 OF 4
AD
A072917


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Am2901

Four-Bit Bipolar Microprocessor Slice

Distinctive Characteristics

- 16-word x 4-bit two-port RAM.
- High speed ALU.
- 9-bit microinstruction word.
- Advanced low-power Schottky processing.
- Four-bit slice cascadable to any number of bits with full carry look-ahead.
- Three-state outputs.
- Shift left, no shift, or shift right entry into RAM from ALU.
- Output multiplexer for direct RAM A-port access or ALU output.
- Status flags include carry-out, sign-bit (negative), overflow and zero detect.
- Four-bit Q-register for scratch pad or accumulator extension.
- Direct ALU entry to Q-register.
- Shift Q-register left or right.
- RAM-shift and Q-shift are easily cascadable.

GENERAL DESCRIPTION

The four-bit bipolar microprocessor slice is designed as a high-speed cascadable element intended for use in CPU's, peripheral controllers, programmable microprocessors and numerous other applications. The microinstruction flexibility of the Am2901 will allow efficient emulation of almost any digital computing machine.

The device, as shown in the block diagram below, consists of a 16-word by 4-bit two-port RAM, a high-speed ALU, and the associated shifting, decoding and multiplexing circuitry. The nine-bit microinstruction word is organized into three groups of three bits each and selects the ALU source operands, the ALU function, and the ALU destination register. The microprocessor is cascadable with full look-ahead or with ripple carry, has three-state outputs, and provides various status flag outputs from the ALU. Advanced low-power Schottky processing is used to fabricate this 40-lead LSI chip.

MICROPROCESSOR SLICE BLOCK DIAGRAM

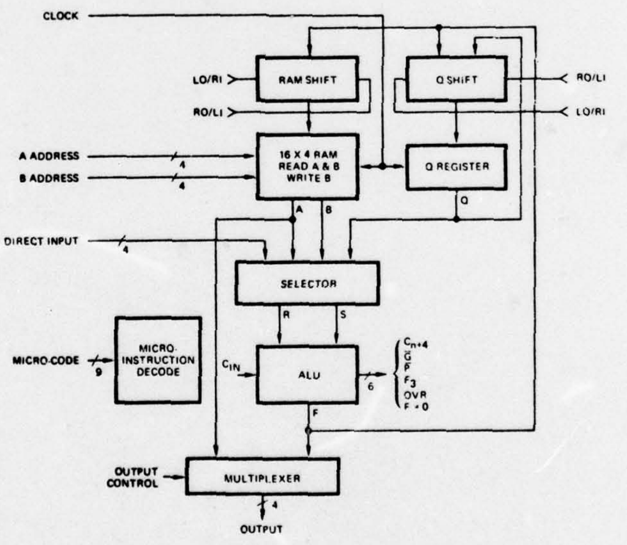
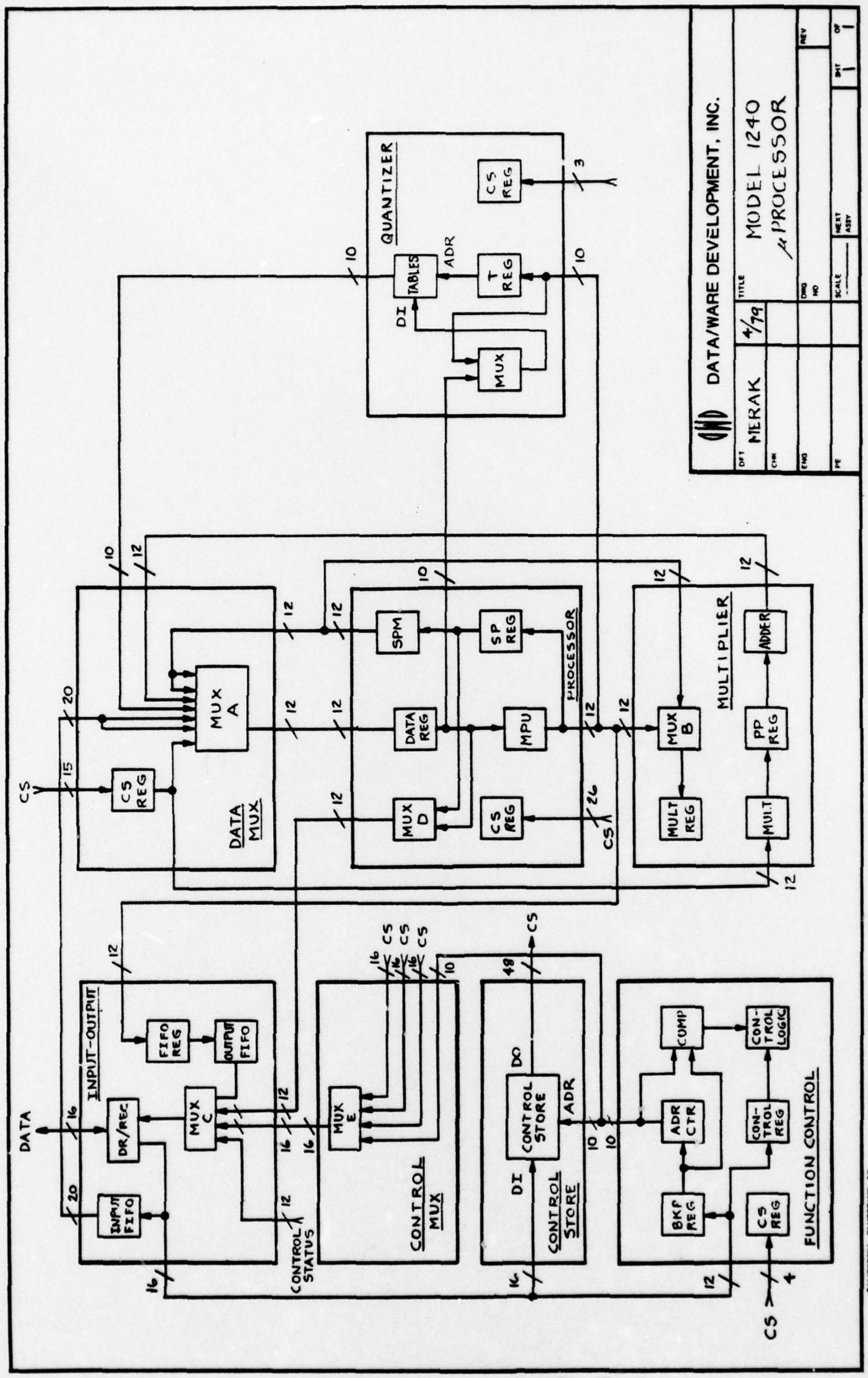


Figure 35. Am 2901 Block Diagram



PRECEDING PAGE NOT FILMED
BLANK

6.1 Functional Description

The major functional elements of the 1240 microprocessor have been partitioned onto individual Plug-In-Boards (PIB). These boards consist of:

1. Input/Output
2. Control Mux
3. Function Control
4. Control Store
5. Data Mux
6. Processor
7. Multiplier
8. Quantizer

The block diagram (Figure 36) shows these boards and their functional relationships in the Model 1240 system.

6.1.1 Input/Output Board

The Input/Output (I/O) board provides the means for transfering and buffering data to and from the MP, typically from the host computer. The microprocessor then appears as a specialized peripheral to the computer. Figure 37. is a block diagram of the I/O Board. Four address lines, A04-A01, enter the Address Receiver where they are used to address registers and to activate functions within the MP. These addresses and functions are listed in Table 6.

The Control Driver/Receiver sends two signals to the interface. These signals, IFIR and OFOR, indicate the input FIFO input or the output FIFO output is ready, respectively. The following control signals are received by the MP:

1. INIT (Initiate) which leads to a master reset of the MP.
2. SIIF (Shift In Input FIFO) which transfers data into the Input FIFO.
3. SOOF (Shift Out Output FIFO) which causes the output FIFO to transfer data out.

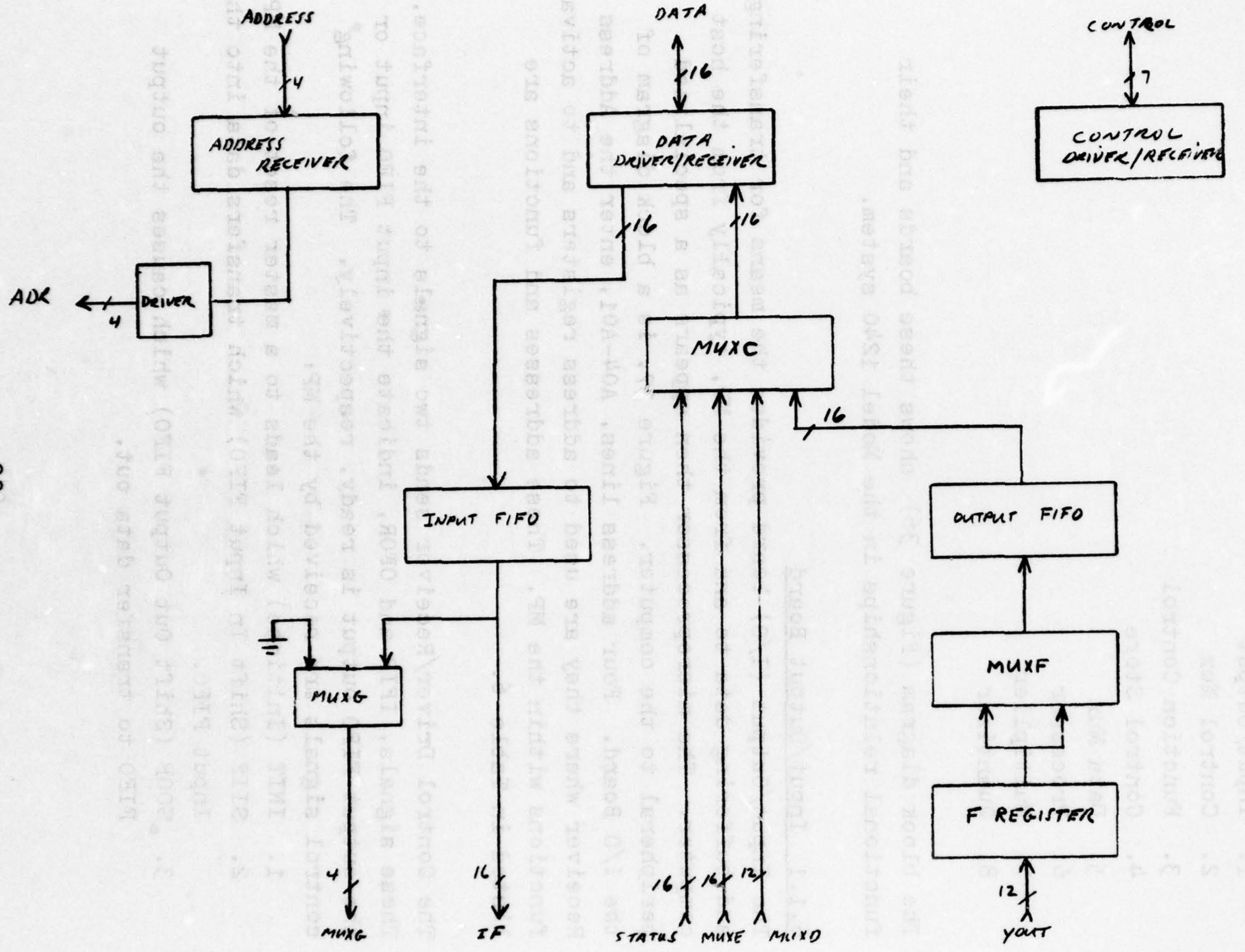


Figure 37. Input/Output Block Diagram

TABLE 6

MICROPROCESSOR ADDRESSABLE REGISTERS

ADDRESS				DESCRIPTION	
MSB			LSB	DATA TO MP (WRITE)	DATA FROM MP (READ)
0	0	0	0	RESET ⚠	Data Register
0	0	0	1	HALT ⚠	
0	0	1	0		Scratch Pad Register
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1	Input FIFO	Output FIFO
1	0	0	0	START ⚠	
1	0	0	1	Control Status Reg.	Control Status Reg.
1	0	1	0		
1	0	1	1	Breakpoint Reg.	
1	1	0	0	Address Counter	Address Counter
1	1	0	1	Control Store 1 (Bits 15-0)	Control Store 1 (Bits 15-0)
1	1	1	0	Control Store 2 (Bits 31-16)	Control Store 2 (Bits 31-16)
1	1	1	1	Control Store 3 (Bits 47-32)	Control Store 3 (Bits 47-32)

⚠ **Addressing of these locations executes the function specified. They are not registers.**

4. DATAMP (Data to the MP) which controls the direction of the bidirectional Data Driver/Receiver.
5. SSTB (Strobe) which strobes data into the MP registers.

The Data Driver/Receiver comprise a 16-bit bidirectional transceiver for transfering data to and from the interface.

The input FIFO is a 64 word X 16-bit first-in/first-out storage element which buffers the data received from the interface. This FIFO together with the MUXG provides either 8- or 12-bit words for the MP to process. If 12-bit words are to be processed, the 12 least significant bits of the FIFO are sent to the Data Mux board. If 8-bit words are to be processed, then either the left or right half word of the FIFO together with four leading zeros are sent to the Data Mux board.

MUXC selects one of four inputs (refer to Table 7) and sends them to the interface via the Data Driver/Receiver. It can select status of the MP (refer to Section 6.2), MUXE from the Control Mux board, MUXD from the Processor board, or the output FIFO.

The Output FIFO is a 64 word X 16 bit storage element which buffers data from the MP. Data from the Processor board (YOUT) is clocked into the F Register and then aligned through MUXF. When 12-bit words are loaded, they are right justified with zero fill. If 8-bit words are loaded they are loaded into either the left or right half word.

6.1.2 Control MUX Board

The Control MUX board provides 16-bit data (MUXE) to the MUXC of the I/O board. It is a 16-bit wide, 4:1 MUX which selects one of three groups of 16 bits each from the Control Store board, or the Control Store Address (CSA) which comes from the Function Control Board. (Refer to Table 7.)

TABLE 7

DATA FROM MP FORMAT

		MUXC			
INTERFACE ADDRESS A3-A0		00XX	01XX	10XX	11XX
(MS)	15	1	FIFO 15	*	MUXF 15
	14	1	" 14	*	" 14
	13	1	" 13	*	" 13
	12	1	" 12	*	" 12
	11	MUXD 11	" 11	WAITIF	" 11
	10	" 10	" 10	WAITOF	" 10
OUTPUT	9	" 09	" 09	HALT	" 09
	8	" 08	" 08	AEQB	" 08
BITS	7	" 07	" 07	BKP	" 07
	6	" 06	" 06	AMOVF	" 06
	5	" 05	" 05	STEP	" 05
	4	" 04	" 04	RIBKP	" 04
	3	" 03	" 03	ENBKP	" 03
	2	02	" 02	$\overline{ENCTR}$	" 02
	1	" 01	" 01	OFMB	" 01
(LS)	0	" 00	" 00	IFMB	" 00

		MUXD		MUXE			
INTERFACE ADDRESS A1,A0		00	10	00	01	10	11
(MS)	15	—	—	1	CS 15	CS 31	CS 47
	14	—	—	AEQB	" 14	" 30	" 46
	13	—	—	$\overline{SEL3}$	" 13	" 29	" 45
	12	—	—	$\overline{SEL2}$	" 12	" 28	" 44
	11	DOUT 11	SOUT 11	$\overline{SEL1}$	" 11	" 27	" 43
	10	" 10	" 10	$\overline{SEL0}$	10	26	" 42
OUTPUT	9	" 09	" 09	CSA 09	" 09	" 25	" 41
	8	" 08	" 08	" 08	08	" 24	40
BITS	7	" 07	" 07	" 07	07	23	39
	6	" 06	" 06	" 06	06	" 22	38
	5	" 05	" 05	" 05	05	21	37
	4	" 04	" 04	04	04	" 20	36
	3	" 03	" 03	03	03	19	35
	2	" 02	" 02	" 02	02	18	34
	1	" 01	" 01	01	01	17	33
(LS)	0	" 00	" 00	10	00	16	32

* RESERVED FOR INTERFACE

6.1.3 Function Control Board

The block diagram for the Function Control board is shown in Figure 38. Control Store bits 18-15 are latched in the Control Store Register for each microinstruction cycle. These bits are decoded to provide the signals that select the register in which the output of the Processor board (YOUT) is to be stored

The Control Logic provides the sequencing of the MP. This is accomplished using a master clock (MCLK) which generates eight overlapping phase clocks for each microinstruction cycle. These phase clocks and their relationship to a microinstruction cycle are shown in Figure 39. The phase clocks are used throughout the MP to produce edges and pulses for the sequential operation of the elements of the MP.

In most cases, the leading edge of T0 clocks the results of the completed cycle into the pipeline register and the leading edge of T1 clocks the next microinstruction into the Control Store Registers. The primary clock functions are distributed as follows:

T0 - a) The leading edge loads the Data Register of the Processor board every cycle unless inhibited by Control Store bits 27-25.

b) The leading edge loads the SP Register of the Processor board if specified by Control Store bits 18-16.

c) The leading edge loads the Mult Register of the Multiplier board if specified by Control Store bits 18-16 or 15.

d) The leading edge loads the PP Register of the Multiplier board every cycle.

e) The leading edge loads the QT Register of the Quantizer board if specified by Control Store bits 18-16.

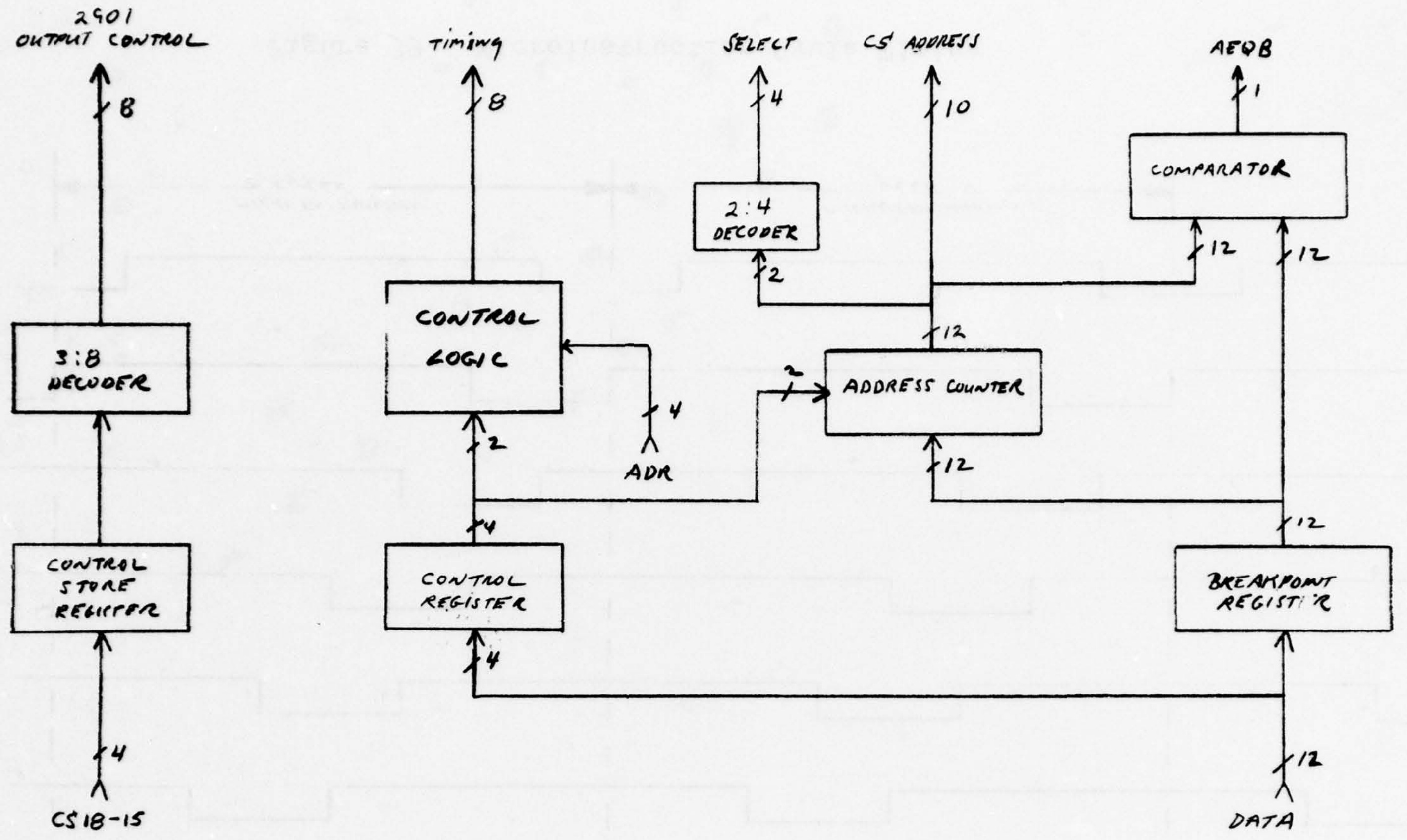


Figure 38. Function Control Block Diagram

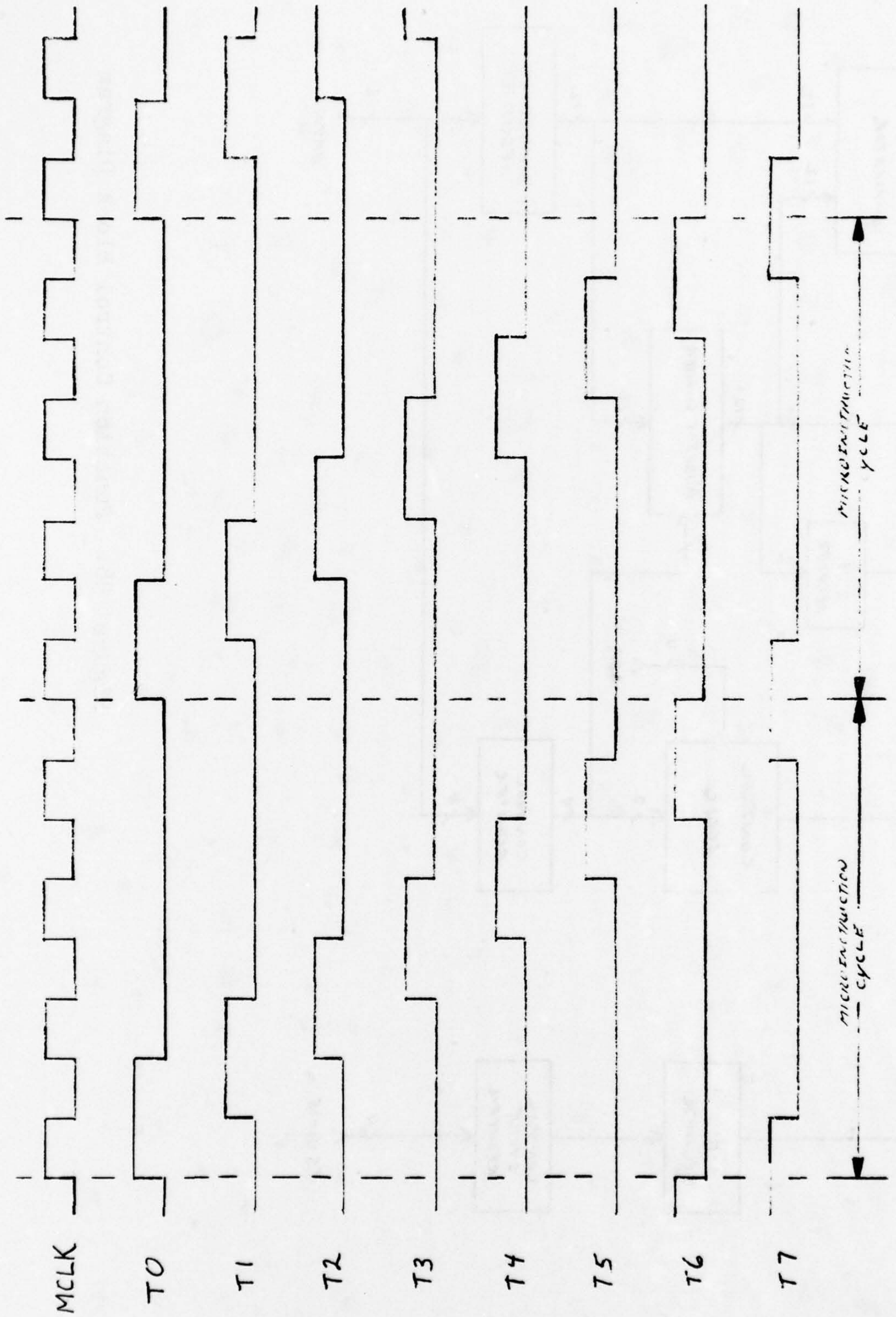


Figure 39. Microinstruction Cycle Timing

T1 - a) The leading edge loads the Control Store Register every cycle time.

b) The pulse shifts out data from the input FIFO of the I/O board if specified by Control Store bits 27-25.

T3 - a) The pulse shifts in data to the output FIFO of the I/O board if specified by Control Store bits 18-16.

b) The pulse writes data into the Scratch Pad Memory (SPM) of the Processor board if specified by Control Store bit 47.

T5 - a) The pulse writes the Quantizer Table (QT) of the Quantizer board if specified by Control Store bits 18-16.

T6 - a) The clock is used to control the Microprocessor Unit (MPU) of the Processor board.

The MP can be halted by either a reset or a halt function. There are two types of resets:

1. System Reset:
   Generated by the computer system to which the MP is interfaced, it usually acts as a system master reset.
2. Program Reset:
   Generated by writing into the MP addressable register location 0000 (refer to Table 6).

The halt is generated by the host computer writing into the addressable register location 0001 of the MP. If halted, the MP can be started by writing into the MP addressable register location 1000 (refer to Table 6). Once started the MP will begin executing microinstructions until either halted or a wait condition occurs. If the "step" bit is set in the Control Register, the MP will execute only a single microinstruction following each start.

The MP will "wait" if the microinstruction to be executed attempts to read from an empty input FIFO or write to a full output FIFO. When the FIFO causing the wait condition is again ready, the MP will continue execution.

Four LED indicators on the Function Control board identify when the MP is in an execution, wait, halt, and/or breakpoint condition.

The Control Register is loaded with data bits 5-2 when the Control Status Register is loaded. These bits control the step mode, jump condition, Breakpoint enable, and Address Counter disable. The step bit, if set, causes the MP to execute a single instruction following each start function. The jump control bit determines whether to jump to address zero or the address contained in the Breakpoint Register when the microinstruction specifies a jump operation. Breakpoint enable, if set, causes the MP to halt when the contents of the Address Counter are equal to the contents of the Breakpoint Register. The Address Counter disable, if set, inhibits the Address Counter from incrementing. All control and status bits are discussed more fully in Section 6.2.

The Address Counter is a 12-bit counter which can address up to four Control Store Boards. The 10 least significant bits are sent to all four boards and select one of 1K words. Decoding of the two most significant bits provides four select signals, one for each board. The Address Counter can be preset from the data lines as an MP Register (address 1100). The counter is incremented after each microinstruction cycle unless the Address Counter disable bit is set or a jump operation occurs. If a jump occurs and the jump bit is not set, the next address will be at location zero. If the jump bit is set, the next location will be that contained in the Breakpoint Register. When the counter disable bit is set, the address will remain unchanged. A reset will set the counter to zero. Presetting of the Address Counter affects the contents of the Breakpoint

Register so that, following a counter preset, the counter and the Breakpoint Register will contain the same data.

Halting of the MP occurs when the contents of the 12-bit Breakpoint Register equal those of the Address Counter (assuming the breakpoint enable bit is set). Its contents can also serve as a jump address if the jump bit of the CSR is set and the microinstruction specifies a jump operation. For this jump, the contents of the Breakpoint Register are loaded into the Address Counter, thus providing the address of the next microinstruction to be executed. The contents of the Breakpoint Register are unaltered in this operation. This register is loaded from the data lines as an MP register (address 1101).

The Comparator compares the 12 bits of the Address Counter with the 12 bits of the Breakpoint Register and forces AEQB true if they are equal.

6.1.4 Control Store Board

The 1240 *Microprocessor* is capable of addressing up to four Control Store boards. Each board contains 1K words, 48 bits wide. These are the basic microinstructions of the Model 1240 microprocessor. A block diagram for a single Control Store (CS) board is shown in Figure 40. The CSA address and select lines originate from the Address Counter on the Function Control board. The select line enables one of four boards while the address lines select the location to be accessed.

Each board is configured in three 1K X 16 blocks. These blocks are loaded as three independent MP Registers (addresses 1111, 1110, 1101) with the information on the data lines being written into the specified address of the enabled block (WE3, WE2, WE1). During reading of data from the CS, the contents of the address specified is transfered to the interface in 16-bit words via the Control Mux board (refer to Table 6).

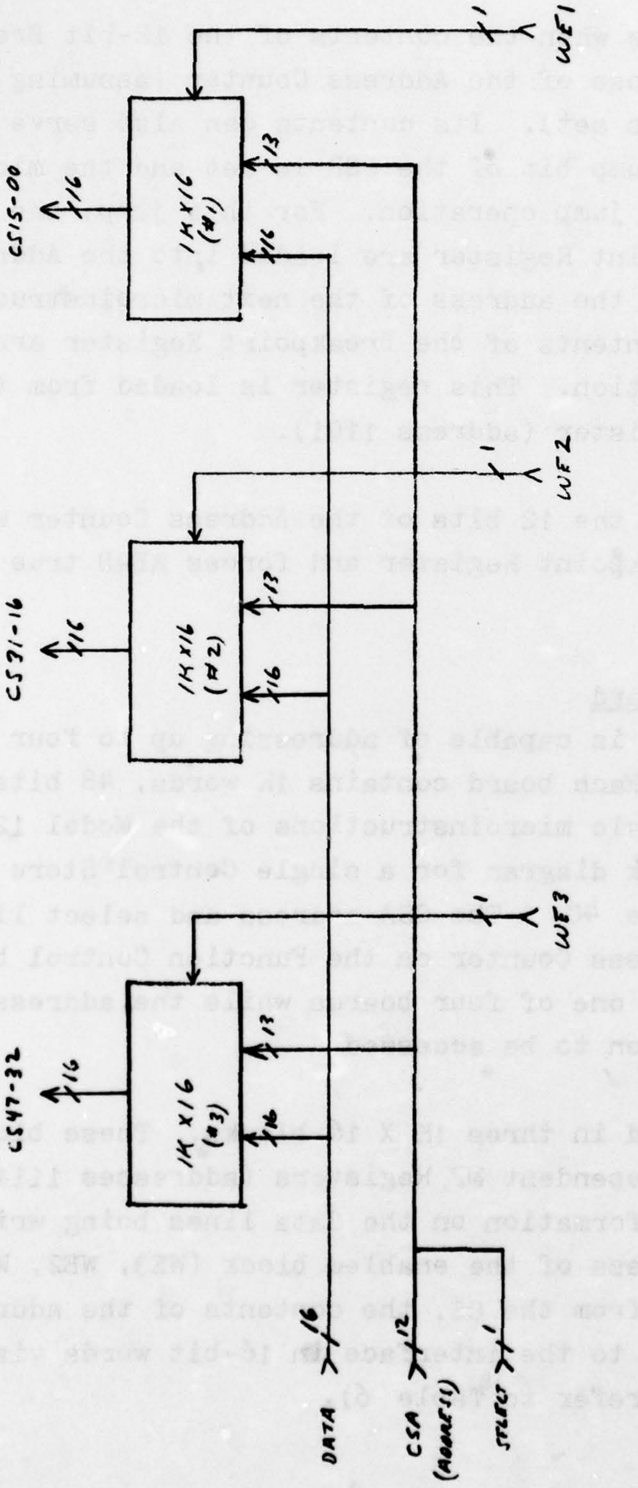


Figure 40. Control Store Block Diagram

During MP execution the contents of the location specified by the Address Counter is placed on the CS47-CS00 lines. This data is latched into the Control Store Register at the beginning of each cycle and becomes the microinstruction for the present instruction cycle.

Refer to Appendix A (T-126-1096), Section 2, for the microinstruction format.

6.1.5 Data Mux Board

The block diagram for the Data Mux board is shown in Figure 41. Control Store bits 27-25 and 11-00 are loaded into the Control Store Register at the beginning of each execution cycle. Bits 27-25 are used to select one of seven input groups to be sent to the Data Register on the Processor board via the 8:1 mux (MUXA). These groups include:

1. Microinstruction Operand (CS11-00)
2. Quantizer Output (Q)
3. Input FIFO right half word (IFLS)
4. Input FIFO left half word (IFMS)
5. Multiplier Output (ADD)
6. Scratch Pad Memory shifted output (SPM)
7. Scratch Pad Memory output (SPM)

The MUXA output format for these groups is shown in Table 8. If CS27, 26 and 25 are all zero the loading of the Data Register is inhibited and its contents remains unchanged.

The 3:8 decoder has three output signals; two to the input FIFO and one to the Data Register. The two to the FIFO shift data out when the FIFO is selected as the input source to the Data Register and the remaining signal enables or disables the loading of the Data Register as required.

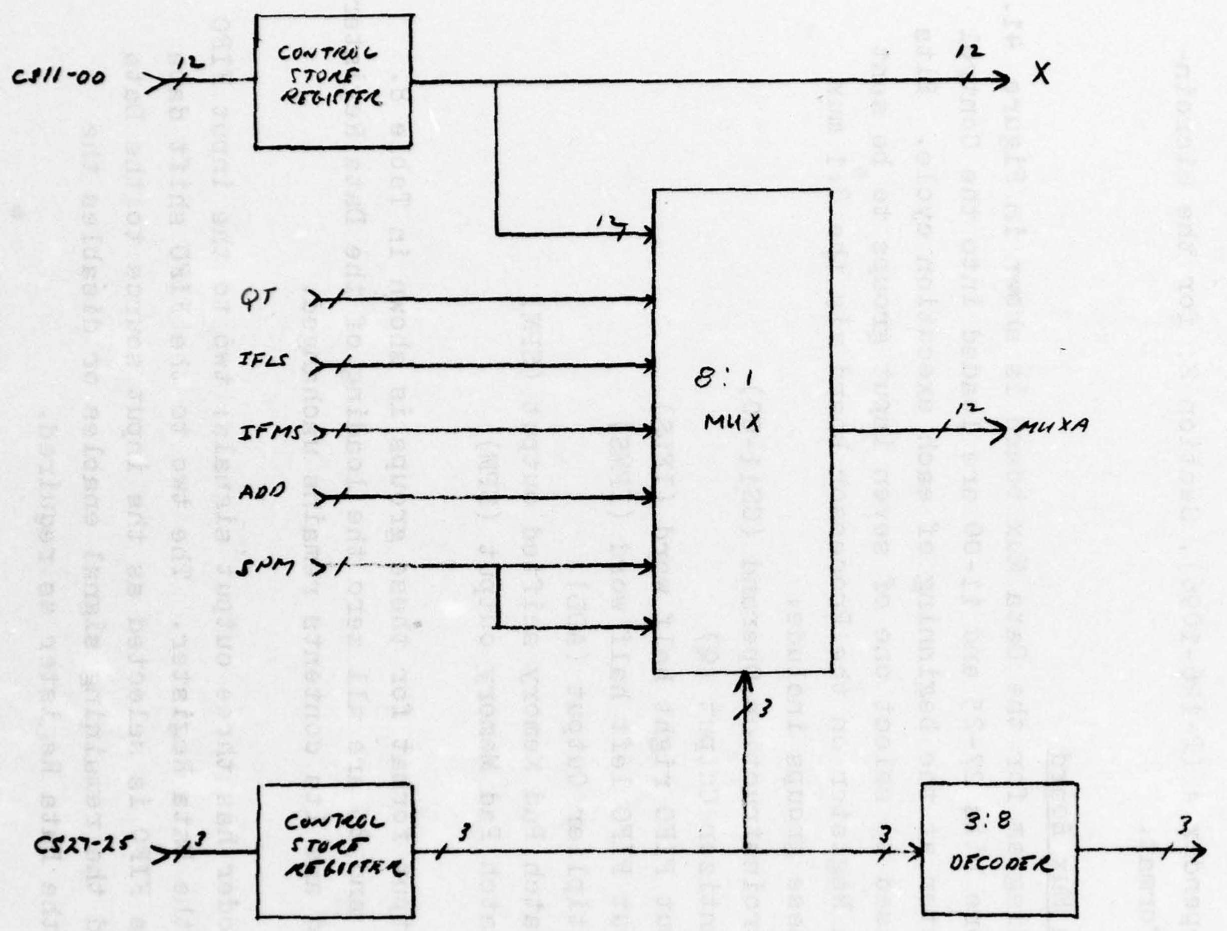


Figure 41. Data Mux Block Diagram

TABLE 8

MUXA OUTPUT FORMAT

		MUXA							
D LOAD (CS 27,26,25)		111	110	101	100	011	010	001	000
(MS)	11	X11	QT11	MUXG3	MUXG3	ADJ11	SPM11	SPM11	
	10	X10	QT10	MUXG2	MUXG2	" 10	" 11	" 10	
	9	X09	QT09	MUXG1	MUXG1	" 09	" 10	" 09	NO
	8	X08	QT08	MUXG0	MUXG0	" 08	" 09	" 08	D
OUTPUT	7	X07	QT07	IF07	IF15	" 07	" 08	" 07	LOAD
	6	X06	QT06	IF06	IF14	" 06	" 07	" 06	
BITS	5	X05	QT05	IF05	IF13	" 05	" 06	" 05	
	4	X04	QT04	IF04	IF12	" 04	05	" 04	
	3	X03	QT03	IF03	IF11	" 03	04	03	
	2	X02	QT02	IF02	IF10	" 02	03	02	
	1	X01	QT01	IF01	IF09	" 01	02	01	
(LS)	0	X00	QT00	IF00	IF08	" 00	01	00	

6.1.6 Processor Board

The Processor board includes the Data Register, Microprocessor Unit (MPU), Scratch Pad Register, Scratch Pad Memory, the MUXD, and the Control Store Register as shown in Figure 42. The Data Register receives selected data from MUXA of the Data Mux board and latches it at the end of each microinstruction cycle. This data is then available for the MPU, MUXD, or the Quantizer Board during the next microinstruction cycle.

The MPU consists of three Advanced Micro Devices Am 2901 four-bit slice microprocessors and a "look ahead carry" generator which speeds up arithmetic operations. The three Am 2901's operate in parallel to provide processing of 12-bit words. The block diagram of the Am 2901 appears in Figure 43 and its instruction set in Figure 44. The output of the Data Register (DOUT) is sent to the Direct Data Input lines of the 2901 and the processed data is sent to the other MP elements via the Am 2901 Y lines (YOUT). The operation of the Am 2901 is controlled by bits 19-24 and 28-39 of the MP microinstruction which is latched in the Control Store Register.

The Scratch Pad Register (SPR) latches the output of the MPU at the end of a cycle if enabled by bits 18-16 of the microinstruction (refer to Attachment A). The SPR output (SOUT) is available to the Scratch Pad Memory and the MUXD.

The Scratch Pad Memory (SPM) is a 64 X 12 RAM which provides additional data storage beyond the 16 general registers. The SPM is addressed by bits 46-41 of the microinstruction and the write enable is controlled by bit 47. When writing into the SPM, the contents of the SPR are stored at the address specified. The output of the Scratch Pad Memory is available as an input to the MP and to the Multiplier.

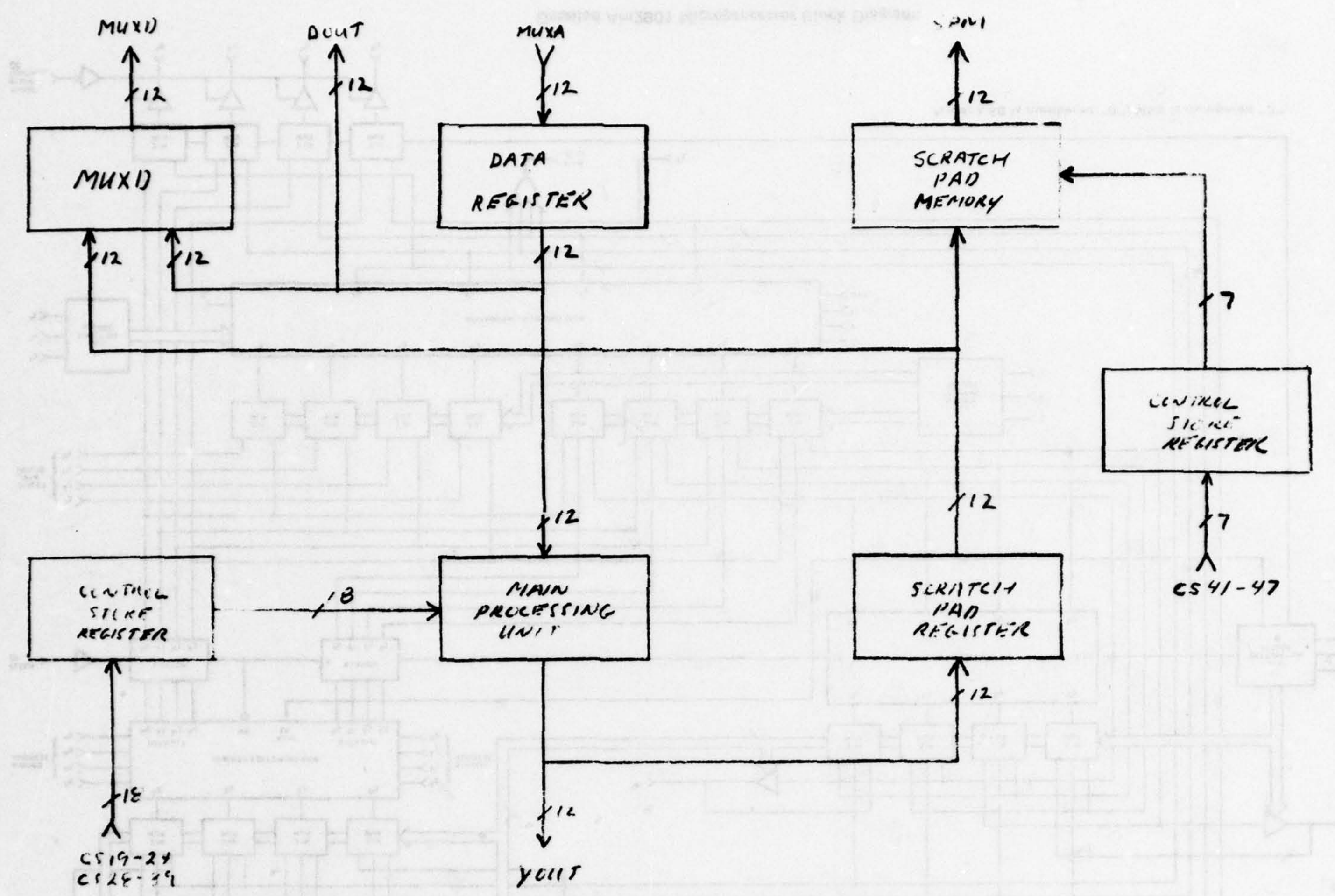
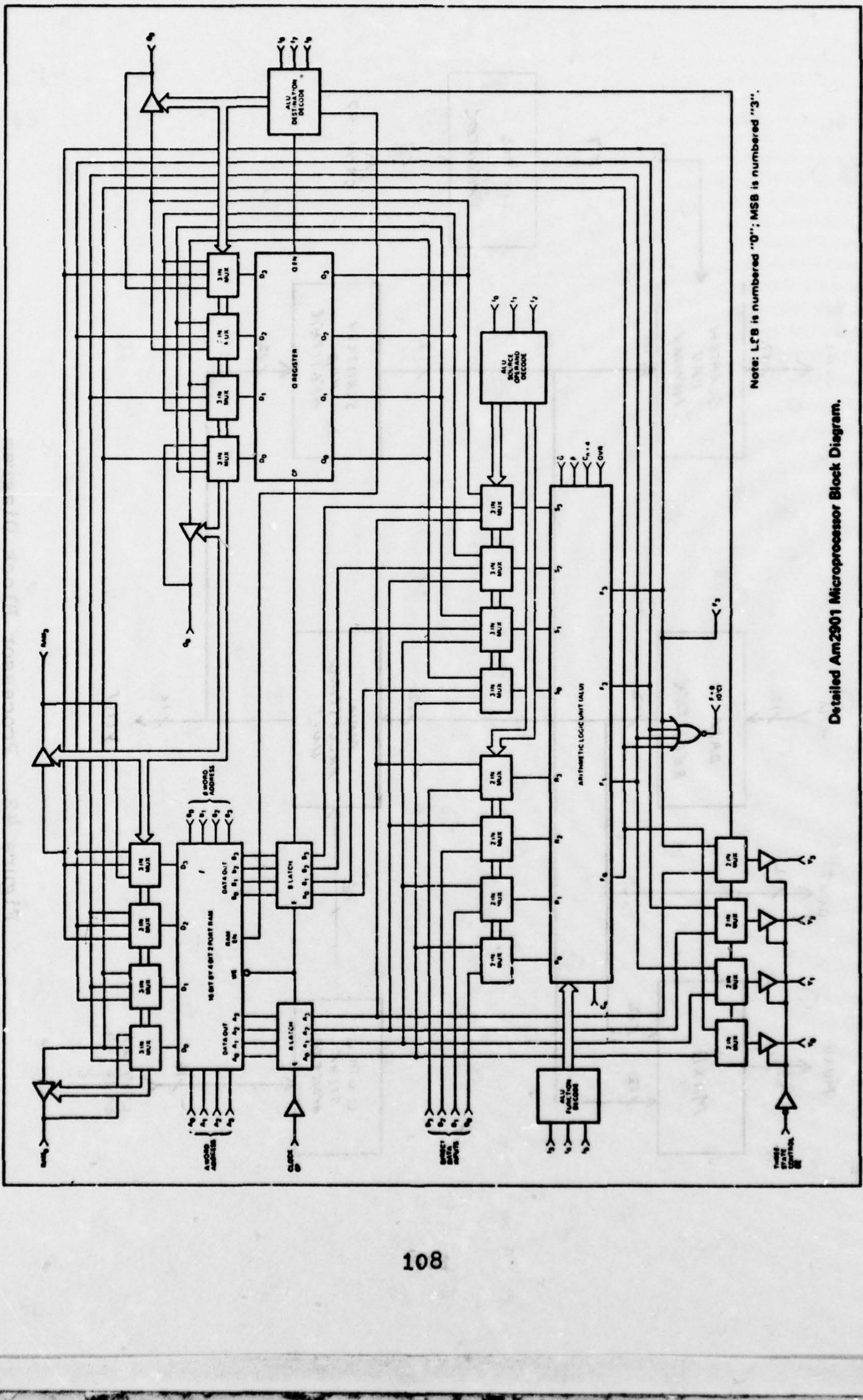


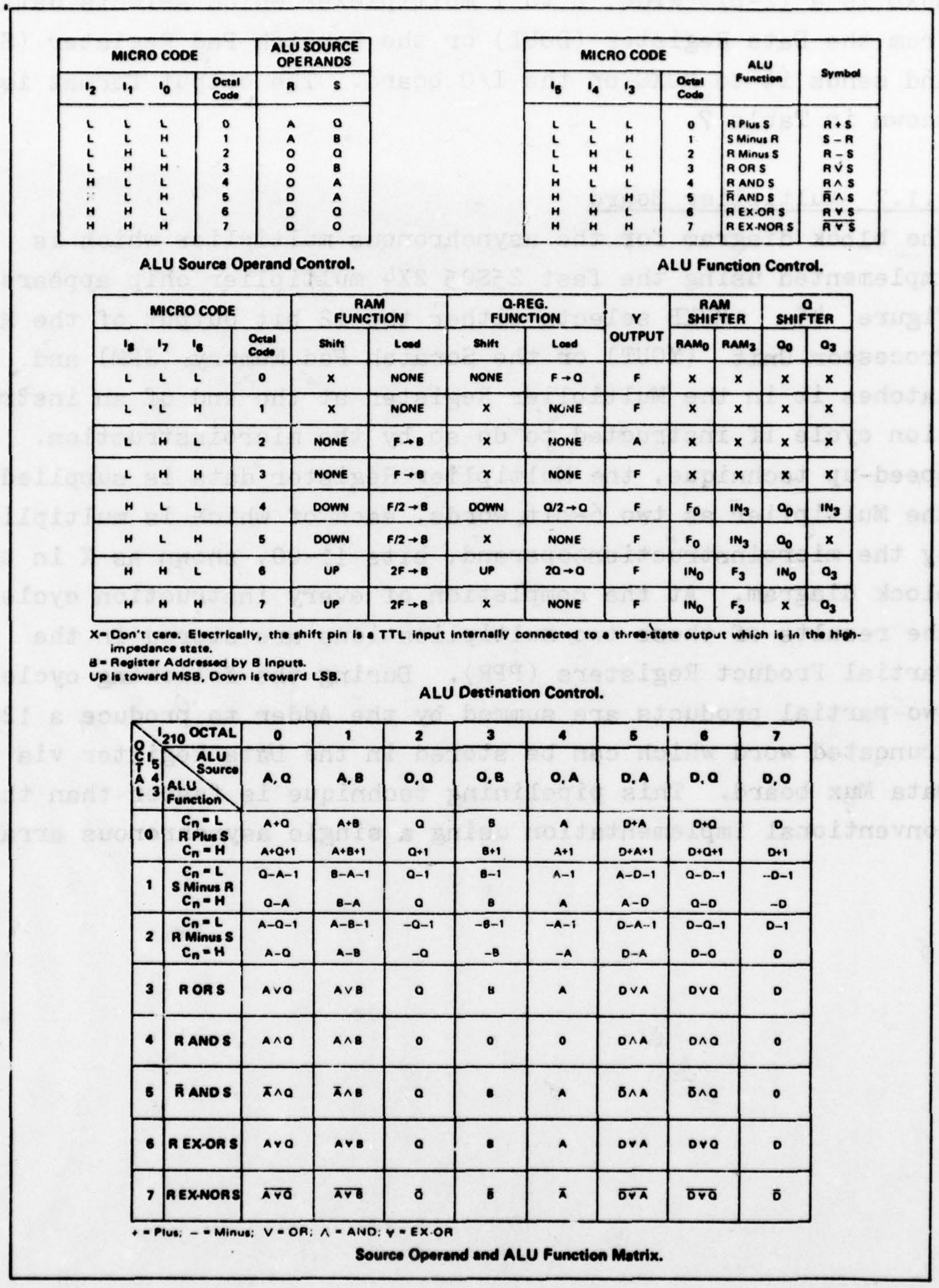
Figure 42. Processor Block Diagram



Note: LSB is numbered "0"; MSB is numbered "3".

Detailed Am2901 Microprocessor Block Diagram.

Figure 43.



MICRO CODE				ALU SOURCE OPERANDS	
I_2	I_1	I_0	Octal Code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	O	Q
L	H	H	3	O	B
H	L	L	4	O	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	O

ALU Source Operand Control.

MICRO CODE				ALU Function	Symbol
I_5	I_4	I_3	Octal Code		
L	L	L	0	R Plus S	$R + S$
L	L	H	1	S Minus R	$S - R$
L	H	L	2	R Minus S	$R - S$
L	H	H	3	R OR S	$R \vee S$
H	L	L	4	R AND S	$R \wedge S$
H	L	H	5	$\bar{R}$ AND S	$\bar{R} \wedge S$
H	H	L	6	R EX-OR S	$R \veebar S$
H	H	H	7	R EX-NOR S	$\overline{R \veebar S}$

ALU Function Control.

MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I_8	I_7	I_6	Octal Code	Shift	Load	Shift	Load		RAM_0	RAM_3	Q_0	Q_3
L	L	L	0	X	NONE	NONE	$F \rightarrow Q$	F	X	X	X	X
L	L	H	1	X	NONE	X	NONE	F	X	X	X	X
L	H	L	2	NONE	$F \rightarrow B$	X	NONE	A	X	X	X	X
L	H	H	3	NONE	$F \rightarrow B$	X	NONE	F	X	X	X	X
H	L	L	4	DOWN	$F/2 \rightarrow B$	DOWN	$Q/2 \rightarrow Q$	F	F_0	IN_3	Q_0	IN_3
H	L	H	5	DOWN	$F/2 \rightarrow B$	X	NONE	F	F_0	IN_3	Q_0	X
H	H	L	6	UP	$2F \rightarrow B$	UP	$2Q \rightarrow Q$	F	IN_0	F_3	IN_0	Q_3
H	H	H	7	UP	$2F \rightarrow B$	X	NONE	F	IN_0	F_3	X	Q_3

X= Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state.

B = Register Addressed by B inputs.

Up is toward MSB, Down is toward LSB.

ALU Destination Control.

OCTAL I_{543}	I_{210} OCTAL	0	1	2	3	4	5	6	7
	ALU Function \ ALU Source	A, Q	A, B	O, Q	O, B	O, A	D, A	D, Q	D, O
0	R Plus S, C_n = L	A+Q	A+B	Q	B	A	D+A	D+Q	D
	R Plus S, C_n = H	A+Q+1	A+B+1	Q+1	B+1	A+1	D+A+1	D+Q+1	D+1
1	S Minus R, C_n = L	Q−A−1	B−A−1	Q−1	B−1	A−1	A−D−1	Q−D−1	−D−1
	S Minus R, C_n = H	Q−A	B−A	Q	B	A	A−D	Q−D	−D
2	R Minus S, C_n = L	A−Q−1	A−B−1	−Q−1	−B−1	−A−1	D−A−1	D−Q−1	D−1
	R Minus S, C_n = H	A−Q	A−B	−Q	−B	−A	D−A	D−Q	D
3	R OR S	$A \vee Q$	$A \vee B$	Q	B	A	$D \vee A$	$D \vee Q$	D
4	R AND S	$A \wedge Q$	$A \wedge B$	0	0	0	$D \wedge A$	$D \wedge Q$	0
5	$\bar{R}$ AND S	$\bar{A} \wedge Q$	$\bar{A} \wedge B$	Q	B	A	$\bar{D} \wedge A$	$\bar{D} \wedge Q$	0
6	R EX-OR S	$A \veebar Q$	$A \veebar B$	Q	B	A	$D \veebar A$	$D \veebar Q$	D
7	R EX-NOR S	$\overline{A \veebar Q}$	$\overline{A \veebar B}$	$\bar{Q}$	$\bar{B}$	$\bar{A}$	$\overline{D \veebar A}$	$\overline{D \veebar Q}$	$\bar{D}$

+ = Plus; − = Minus; $\vee$ = OR; $\wedge$ = AND; $\veebar$ = EX-OR

Source Operand and ALU Function Matrix.

Figure 44. Am2901 Instruction Set

MUXD is a 12-bit wide, 2 to 1 multiplexer which selects data from the Data Register (DOUT) or the Scratch Pad Register (SOUT) and sends it to MUXC on the I/O board. The output format is shown in Table 7.

6.1.7 Multiplier Board

The block diagram for the asynchronous multiplier which is implemented using the fast 25S05 2X4 multiplier chip appears in Figure 45. MUXB selects either the 12 bit output of the Micro-Processor Unit (YOUT) or the Scratch Pad Memory (SPM) and latches it in the Multiplier Register at the end of an instruction cycle if instructed to do so by the microinstruction. As a speed-up technique, the Multiplier Register data is supplied to the Multiplier as two 6-bit words, each of which is multiplied by the microinstruction operand, bits 11-00, shown as X in the block diagram. At the completion of every instruction cycle the results of these two multiplications are stored in the Partial Product Registers (PPR). During the following cycle the two partial products are summed by the Adder to produce a 12-bit, truncated word which can be stored in the Data Register via the Data Mux board. This pipelining technique is faster than the conventional implementation using a single asynchronous array.

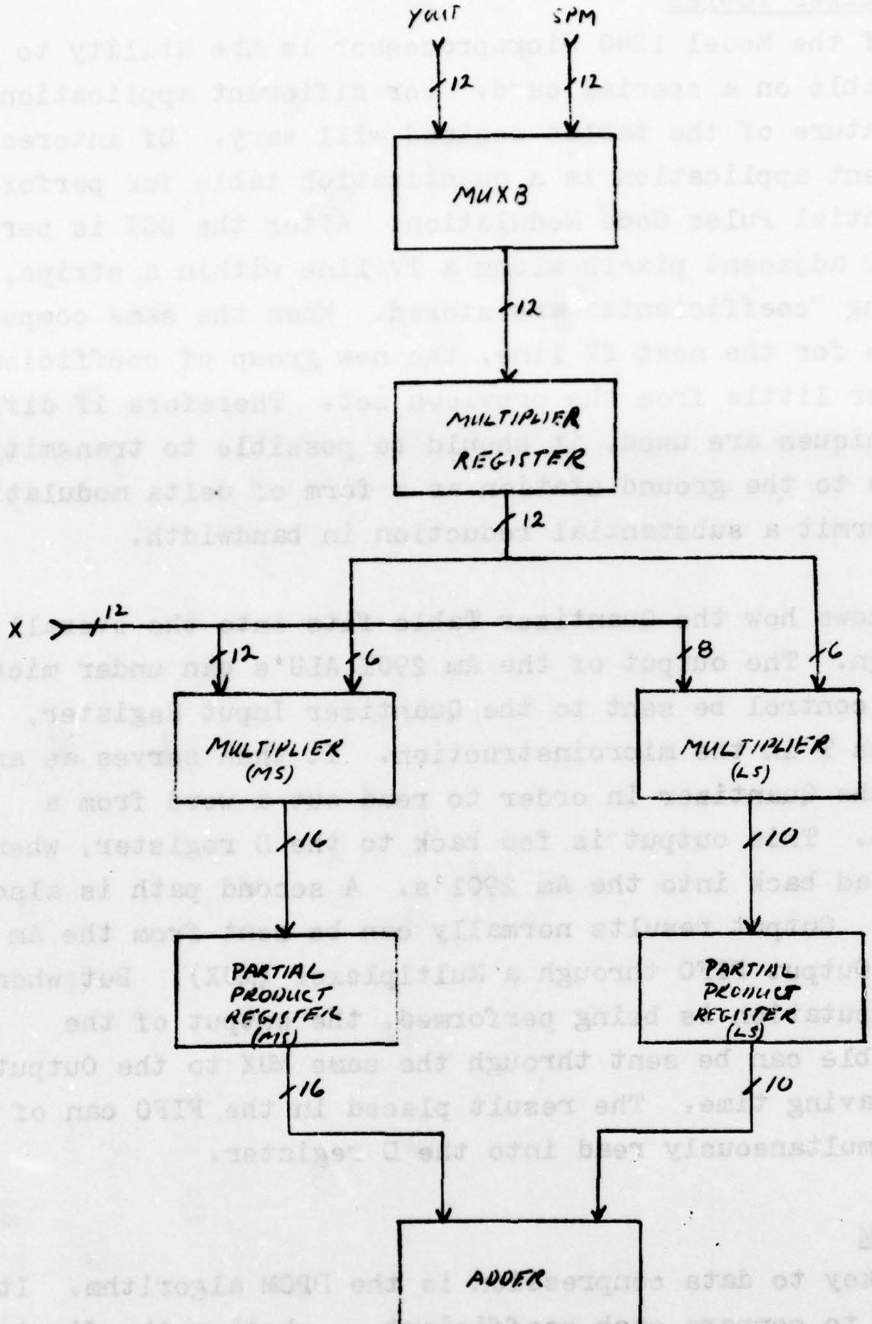


Figure 45. Multiplier Block Diagram

6.1.8 Quantizer Tables

A feature of the Model 1240 Microprocessor is the ability to address a table on a special card. For different applications the exact nature of the tables desired will vary. Of interest in the present application is a quantization table for performing Differential Pulse Code Modulation. After the DCT is performed on 32 adjacent pixels along a TV line within a stripe, the resulting "coefficients" are stored. When the same computation is made for the next TV line, the new group of coefficients should differ little from the previous set. Therefore if differencing techniques are used, it should be possible to transmit the coefficients to the ground station as a form of delta modulation. This will permit a substantial reduction in bandwidth.

Fig. 46 shows how the Quantizer Table fits into the overall system design. The output of the Am 2901 ALU's can under microinstruction control be sent to the Quantizer Input Register, designated as T in the microinstruction. It then serves as an address to the Quantizer in order to read out a word from a stored table. This output is fed back to the D register, whence it can be read back into the Am 2901's. A second path is also of interest. Output results normally can be sent from the Am 2901 to the Output FIFO through a Multiplexer (MUX). But when the DPCM computation is being performed, the output of the Quantizer Table can be sent through the same MUX to the Output FIFO, thus saving time. The result placed in the FIFO can of course be simultaneously read into the D register.

6.1.8.1 DPCM

Clearly the key to data compression is the DPCM algorithm. It is necessary to compare each coefficient -- whether the first harmonic, second, etc. -- with the corresponding coefficient from the preceding TV line. These preceding coefficients are kept in memory. It is the difference between these two coefficients which is used to enter the Quantizer Tables. Note that the Quantizer is actually made up of several tables. This is done to take into account the different expected "sizes" of the

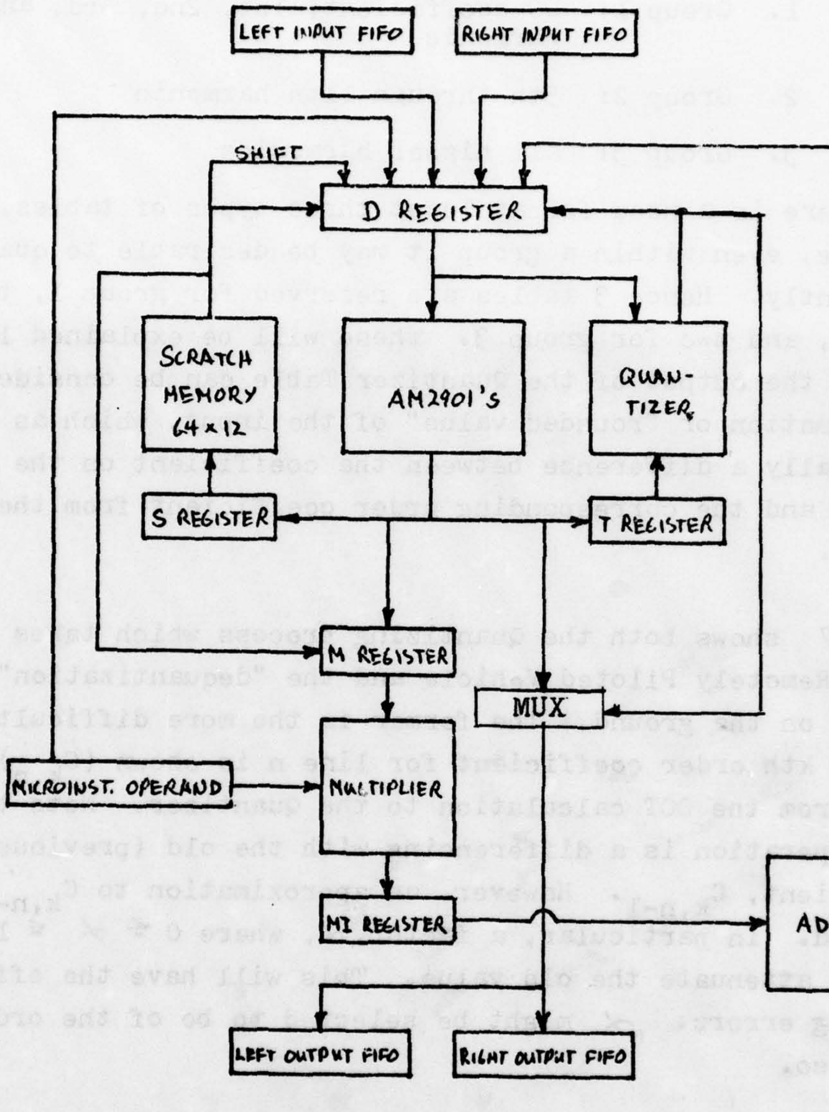


Fig. 46. Microprocessor Logic

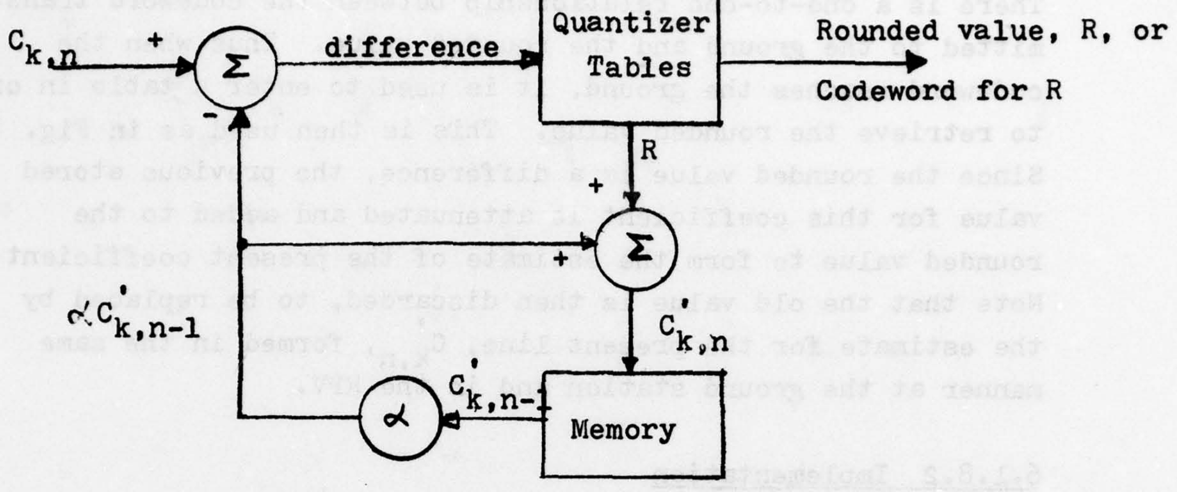
various coefficients. A general rule is that the magnitude of a given coefficient is inversely proportional to its order. Thus the 5th harmonic is expected to be about 1/5th the magnitude of the 1st harmonic. A rough grouping is then possible:

1. Group 1: DC coefficient, 1st, 2nd, 3rd, and 4th harmonic
2. Group 2: 5th through 11th harmoni
3. Group 3: all higher harmonics

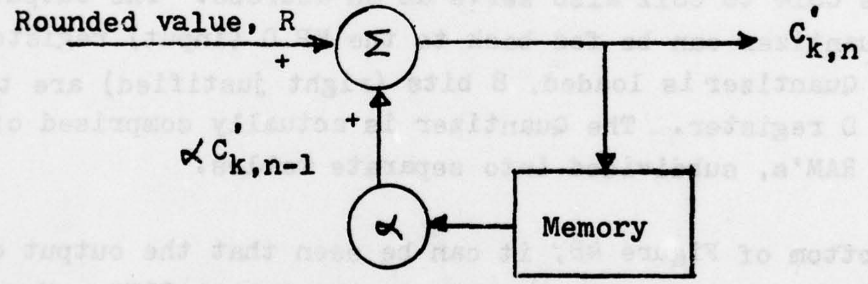
Thus there is a need for at least three types of tables. Furthermore, even within a group it may be desirable to quantize differently. Hence 3 tables are reserved for group 1, two for group 2, and two for group 3. These will be explained later. Briefly the output of the Quantizer Table can be considered an approximation or "rounded value" of the input, which as was noted is actually a difference between the coefficient on the present TV line and the corresponding order coefficient from the preceding TV line.

Fig. 47 shows both the Quantizing process which takes place in the Remotely Piloted Vehicle and the "dequantization" which is done on the ground. The former is the more difficult. The current kth order coefficient for line n is shown ($C_{k,n}$) as an input from the DCT calculation to the Quantizer. Note that the first operation is a differencing with the old (previous line) coefficient, $C_{k,n-1}$. However, an approximation to $C_{k,n-1}$ is employed. In particular, a factor, α, where $0 \leq \alpha \leq 1$, is used to attenuate the old value. This will have the effect of removing errors. α might be selected to be of the order of 0.9 or so.

Fig. 47a shows that this old, attenuated value is subtracted from $C_{k,n}$, and the latter difference, after being truncated, enters the Quantizer. The output of the Quantizer will be compressed to a codeword which is 5, 4, 3, 2, 1, or zero bits. Corresponding to this codeword is a rounded value. As an example, if the input to the Quantizer is 26 in binary, the 5-bit quantized output is 27, the 4-bit one is 29, the 3-bit one is 24, the 2-bit



(a) Quantizing



(b) Dequantizing

Fig. 47. Use of Quantization Tables

one is 17, and the 1-bit one is 8. This assumes the input was postive. If it were negative, the sign of the output would be negative also since the table is symmetrical.

There is a one-to-one relationship between the codeword transmitted to the ground and the rounded value. Thus when the codeword reaches the ground, it is used to enter a table in order to retrieve the rounded value. This is then used as in Fig. 47b. Since the rounded value is a difference, the previous stored value for this coefficient is attenuated and added to the rounded value to form the estimate of the present coefficient. Note that the old value is then discarded, to be replaced by the estimate for the present line, $C'_{k,n}$, formed in the same manner at the ground station and in the RPV.

<u>6.1.8.2 Implementation</u>

Additional detail of the Quantizer is shown in Fig. 48. The Quantizer Tables are shown at the top of the page. The address from the Microprocessor (MP) is placed in the T register, but the high-order address bits, which select the particular table, come from the microinstruction currently being executed. Note that bits CS14 to CS12 also serve as an address. The output of the Quantizer can be fed back to the MP D (input) register. When the Quantizer is loaded, 8 bits (right justified) are taken from the D register. The Quantizer is actually comprised of 8 1024-bit RAM's, subdivided into separate tables.

At the bottom of Figure 48, it can be seen that the output of the MP, YOUT, can be sent directly to the Output FIFO. However, there is an alternate path so that the Quantizer output, which is sent to the MP, can also come directly through the MUX H, to the Output FIFO. This is useful for laboratory work in which the data transmitted is not a codeword but rather the rounded value itself. Since there is a one-to-one relationship between the two, no generality is lost and greater flexibility is gained for experimentation. This rounded value, which is the output of the compression process, is passed on to the dequantizing

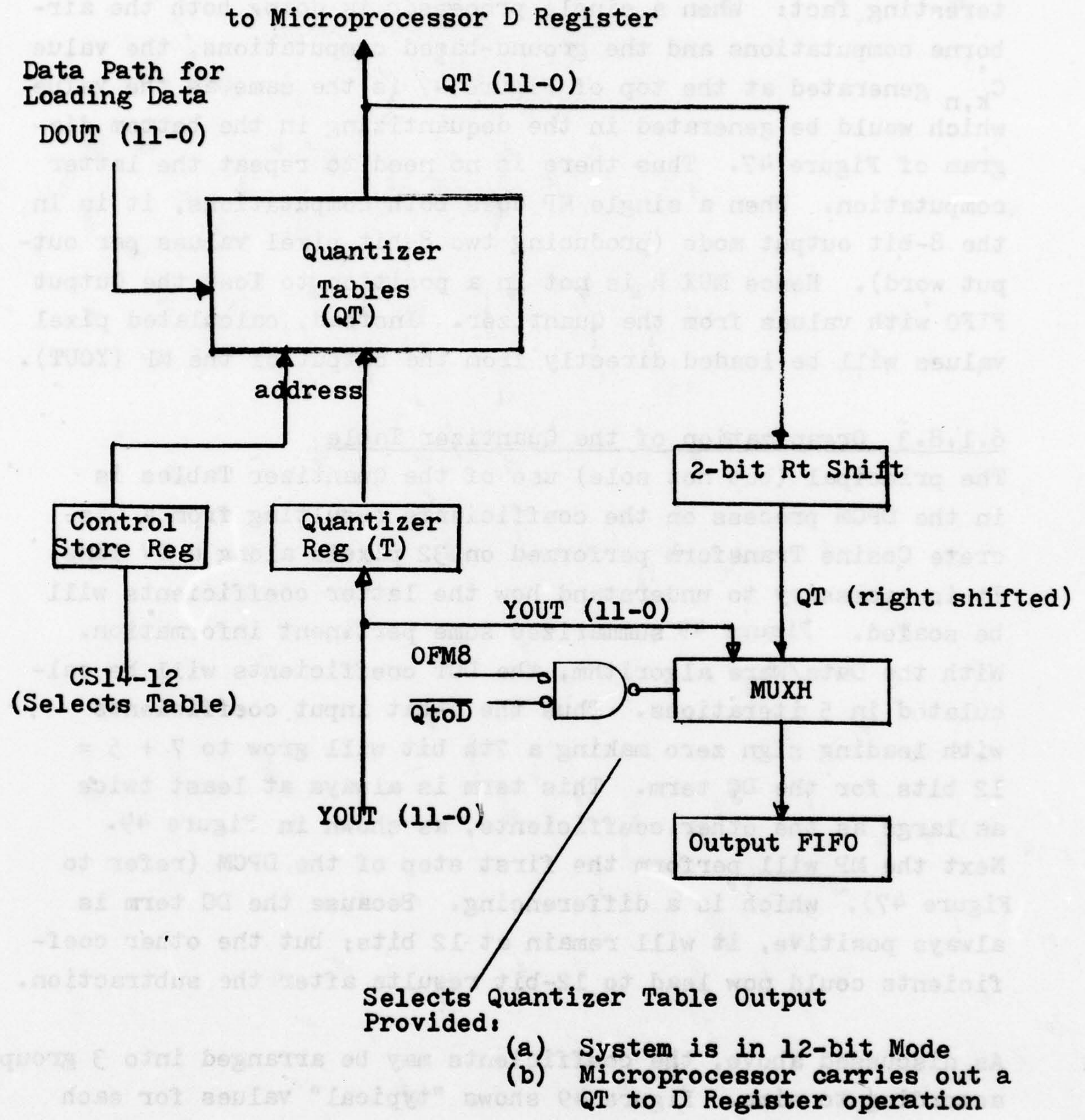


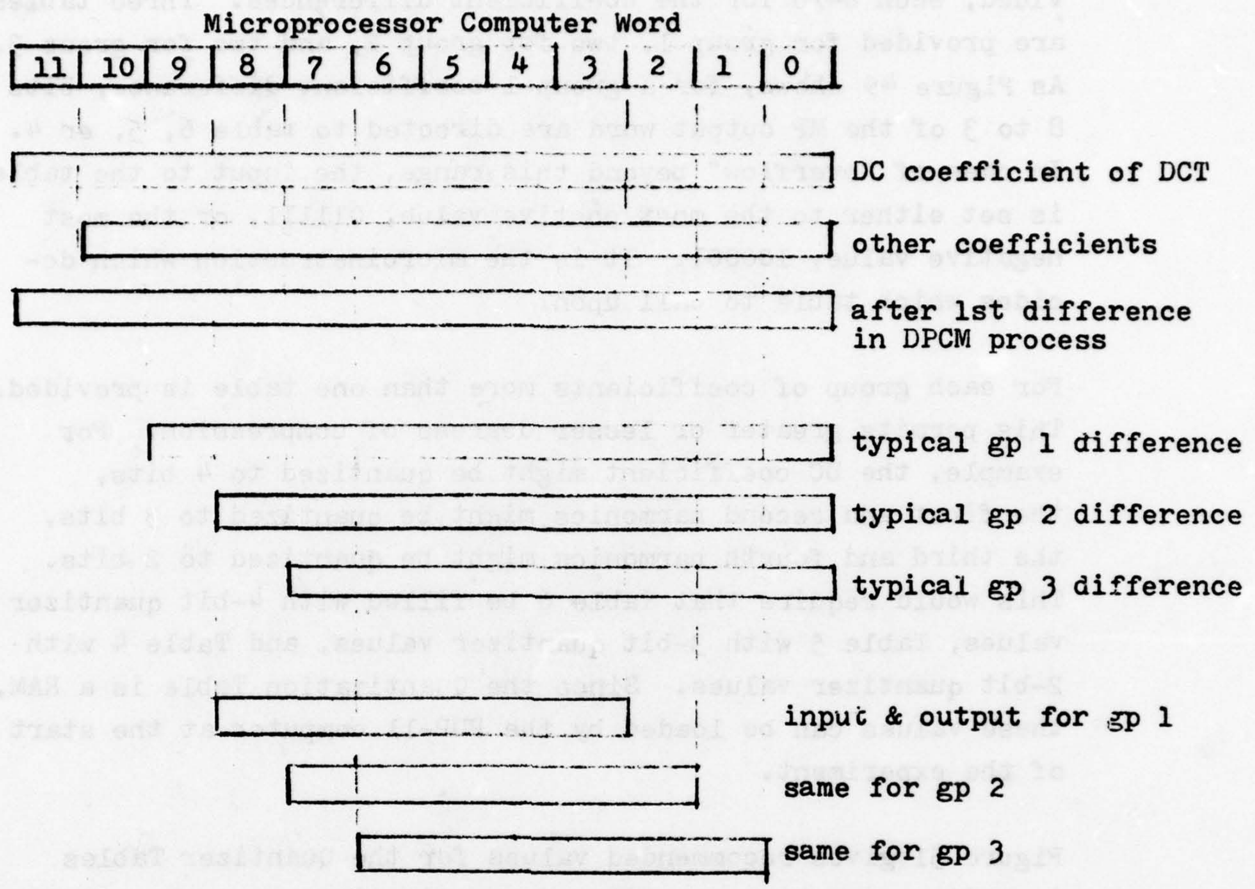
Fig. 48. Loading Output FIFO

process at the simulated ground station. Note the following interesting fact: When a single processor is doing both the airborne computations and the ground-based computations, the value $C'_{k,n}$ generated at the top of Figure 47 is the same as the value which would be generated in the dequantizing in the bottom diagram of Figure 47. Thus there is no need to repeat the latter computation. When a single MP does both computations, it is in the 8-bit output mode (producing two 8-bit pixel values per output word). Hence MUX H is not in a position to load the Output FIFO with values from the Quantizer. Instead, calculated pixel values will be loaded directly from the output of the MP (YOUT).

6.1.8.3 Organization of the Quantizer Table

The principal (but not sole) use of the Quantizer Tables is in the DPCM process on the coefficients resulting from a Discrete Cosine Transform performed on 32 pixels along a TV line. It is necessary to understand how the latter coefficients will be scaled. Figure 49 summarizes some pertinent information. With the Data/Ware algorithm, the DCT coefficients will be calculated in 5 iterations. Thus the 6-bit input coefficients with leading sign zero making a 7th bit will grow to 7 + 5 = 12 bits for the DC term. This term is always at least twice as large as the other coefficients, as shown in Figure 49. Next the MP will perform the first step of the DPCM (refer to Figure 47), which is a differencing. Because the DC term is always positive, it will remain at 12 bits; but the other coefficients could now lead to 12-bit results after the subtraction.

As discussed above, the coefficients may be arranged into 3 groups according to size. Figure 49 shows "typical" values for each group, including the sign bit. The lower part of Figure 49. indicates how 6-bit selections are made from each group to enter the Quantizer. A test is first made to see whether the difference must be "limited", i.e. set to the most positive or most negative value because of overflow beyond the selected 6 bits.



Notes: Starting with positive sign (0) and 6-bit pixels, the resulting DC coefficient and other coeficients will be scaled as shown at top of page (including the sign bit). After the first difference of the DPCM, all differences can be 12 bits.

Calling the first 5 coefficients (including DC, 1st harmonic, 2nd, ..., 4th) group 1, the next 7 coefficients group 2, and the rest group 3, "typical 1st differences" in the DPCM are as shown.

6-bit selections are made for each group in order to enter the Quantization Table. Limiting is first carried out. The Table outputs a 6-bit "rounded value"

Fig. 49. Scaling Considerations for DPCM

Figure 50 shows that there are several different tables provided, each 64x8 for the coefficient differences. Three tables are provided for group 1, two for group 2, and two for group 3. As Figure 49 shows, for a group 1 coefficient difference, bits 8 to 3 of the MP output word are directed to table 6, 5, or 4. In case of "overflow" beyond this range, the input to the table is set either to the most postive value, 011111, or the most negative value, 100001. It is the microinstruction which decides which table to call upon.

For each group of coefficients more than one table is provided. This permits greater or lesser degrees of compression. For example, the DC coefficient might be quantized to 4 bits, the first and second harmonics might be quantized to 3 bits, the third and fourth harmonics might be quantized to 2 bits. This would require that Table 6 be filled with 4-bit quantizer values, Table 5 with 3-bit quantizer values, and Table 4 with 2-bit quantizer values. Since the Quantization Table is a RAM, these values can be loaded by the PDP-11 computer at the start of the experiment.

Figure 51 gives recommended values for the Quantizer Tables depending upon how many bits of quantization are to be permitted. Only positive values are shown in Figure 51, but there is of course an equal amount of space devoted to the negative values so that the tables are symmetrical about zero. The values in the table are the rounded values or approximations to the input difference. For small values of the differences, the steps in the table are rather fine. For larger values, the steps become rather large.

When a difference is sent to tables 6 to 4 by the microinstruction, only bits 8 to 3 of the MP word are used to address the table. Similarly bits 7 to 2 of the MP word are used to address tables 3 and 2; while bits 6 to 1 are used as the address to tables 1 and 0. This serves as an adjustment for the differing expected sizes of the three groups of coefficients and results in a more efficient encoding of the information.

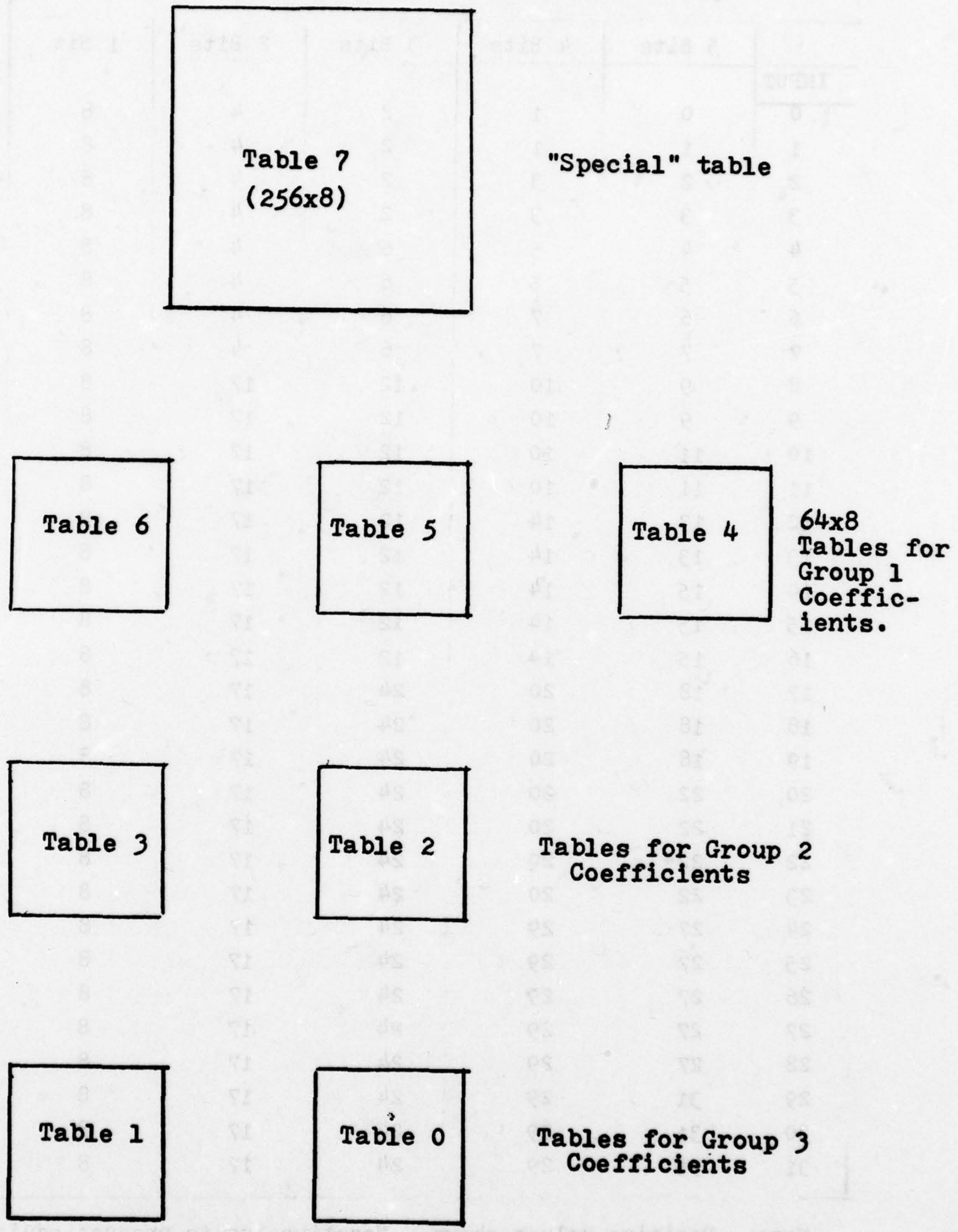


Table 7
(256x8)
"Special" table
Table 6
Table 5
Table 4
64x8 Tables for Group 1 Coefficients.
Table 3
Table 2
Tables for Group 2 Coefficients
Table 1
Table 0
Tables for Group 3 Coefficients

Fig. 50. Detail of Quantization Tables

	Output as a Function of number of Quantization Bits				
INPUT	5 Bits	4 Bits	3 Bits	2 Bits	1 Bit
0	0	1	2	4	8
1	1	1	2	4	8
2	2	3	2	4	8
3	3	3	2	4	8
4	4	5	6	4	8
5	5	5	6	4	8
6	6	7	6	4	8
7	7	7	6	4	8
8	9	10	12	17	8
9	9	10	12	17	8
10	11	10	12	17	8
11	11	10	12	17	8
12	13	14	12	17	8
13	13	14	12	17	8
14	15	14	12	17	8
15	15	14	12	17	8
16	15	14	12	17	8
17	18	20	24	17	8
18	18	20	24	17	8
19	18	20	24	17	8
20	22	20	24	17	8
21	22	20	24	17	8
22	22	20	24	17	8
23	22	20	24	17	8
24	27	29	24	17	8
25	27	29	24	17	8
26	27	29	24	17	8
27	27	29	24	17	8
28	27	29	24	17	8
29	31	29	24	17	8
30	31	29	24	17	8
31	31	29	24	17	8

Note: Positive values shown. Negative inputs produce equivalent negative outputs.

Figure 51. Quantization Table

6.1.8.4 Special Table

The seven small tables for quantizing differences in coefficients during the DPCM operation have been described. An eighth table (Table 7) which is 256x8 in size has been provided for greater generality. It was considered desirable to permit a table look-up on pixel values. Hence an 8-bit pixel value can be fed as input to Table 7 and will produce an output also 8 bits in length. This can be done at extremely high speed as pixels are being generated (perhaps during the inverse transformation) prior to being sent to the frame store memory.

The eight bits sent to the special table are bits 11 to 4 of the MP output. Bits 3 to 0 are discarded. This process is sometimes referred to as the use of a "function memory". Prof. Harry C. Andrews of the University of Southern California has stated, "Probably the simplest and yet most powerful enhancement technique is that known as nonlinear point processes. Such processes refer to the mapping of individual pixels to new values independent of their neighboring pixel values. There are many motivating factors behind the use of point processes for image enhancement. Possibly the most sound one is that of attempting the removal of monotonic nonlinearities experienced during imaging. An example of such might be correction for film gamma in a photographic process. However, some less esoteric but more practical point process motivation exists from the desire to simply use the available gray shades of a display device more effectively than might be indicated by the histogram of the scanned images. Consequently, stretching, noise clipping, and histogram equalization become possibly useful candidates."

Thus all that is required is to place the desired mapping function into the special memory. During the inverse transformation -- after the inverse DPCM and inverse DCT -- the pixels are recovered. At this point they can be sent to the special table to attempt enhancement. There are other possible uses for the tables, but the above at least presents some useful concepts for experimentation.

6.2 Control and Status Register

The Control & Status Register (CSR) is a 16-bit register which can be written into by the host computer for MP control and read from to determine MP status. This register has address 1001 (refer to Table 6). The contents of the CSR are shown in Table 9. Bits 15-12 are reserved for use in defining the interface between the MP and the computer system. In all, the CSR provides six control bits (05-00) and six status bits (11-06).

In writing into the CSR, the information on the data lines are loaded into the control bits while the status bits remain unchanged. The status and control state of the MP can be monitored by reading the CSR.

6.2.1 Control

The following CSR bits control the operation of the MP:

IFM8 (Bit 00):	Controls the word length read from the input FIFO into the Data Register. If set ("1") an 8-bit half word is read and if cleared ("0"), a 12-bit word is read.
OFM8 (Bit 01):	Controls the word length transfered from the output FIFO to the interface via the data lines. If set, an 8-bit half word is transfered and if cleared, a 12-bit word is transfered.
$\overline{\text{ENCTR}}$ (Bit 02):	If set, this bit disables the incrementing of the Address Counter on the Function Control board. If cleared, the counter is enabled and incremented after each instruction cycle. Disabling the Address Counter does not inhibit jump operations.
ENBKP (Bit 03):	If set, causes the MP to halt when the contents of the Breakpoint Register are equal to the contents of the Address Counter. When the halt occurs, the Address Counter

TABLE 9

MP CONTROL STATUS REGISTER

BIT	TERM	TYPE*	LOCATION**	"0"	"1"	DESCRIPTION
15						
14						
13						
12						
11	WAITIF	S	FC	NO	YES	Wait due to input FIFO empty
10	WAITOF	S	FC	NO	YES	Wait due to output FIFO full
09	HALT	S	FC	NO	YES	Halted
08	AEQB	S	FC	NO	YES	Address Counter equal Breakpoint Reg.
07	BKP	S	FC	NO	YES	Halted at Breakpoint
06	AMOVF	S	I/O	NO	YES	Arithmetic overflow
05	STEP	C	FC & I/O	RUN	STEP	Step Mode
04	JUMP	C	FC & I/O	RTZ	RTBKP	Jump Control
03	ENBKP	C	FC & I/O	DISABLE	ENABLE	Enable Breakpoint
02	$\overline{\text{ENCTR}}$	C	FC & I/O	ENABLE	DISABLE	Enable Address Counter
01	OFMB	C	I/O	12-Bit	8-Bit	Output FIFO Mode
00	IFMB	C	I/O	12-Bit	8-Bit	Input FIFO Mode

*C = Control, S = Status
**FC = Function Control Board, I/O = Input/Output Board

Note: Following a reset all CSR bits are in the "0" state.

will have been incremented ahead, and the instruction at the location indicated by the Breakpoint Register will have been executed. If cleared, the MP will not halt when the contents of the Address Counter and Breakpoint Registers are equal.

JUMP (Bit 04): When a microinstruction directs the MP to execute a jump operation, this bit selects one of two addresses for the jump. If set, the next microinstruction following the jump instruction will be at the location specified by the Breakpoint Register, and, if cleared, it will be at location zero.

STEP (Bit 05): If set, this bit places the MP in the STEP mode and, if cleared, the MP will be in the RUN mode. In the RUN mode, the MP executes microinstructions until stopped by a HALT command, a reset, or by encountering a breakpoint. In the step mode, the MP halts after the execution of each microinstruction and can then be "stepped" by a START command. (The HALT and START commands are activated by writing into the respective MP addressable registers (refer to Table 6).

6.2.2 Status

The status of the MP is indicated by the following bits:

AMOVF (Bit 06): When set, indicates that an overflow has occurred in the Main Processing Unit while executing an arithmetic function. A reset is required to clear this bit.

BKP (Bit 07): This bit, when set, indicates that the MP has encountered the breakpoint specified in the Breakpoint Register. This bit cannot be set unless the breakpoint is enabled (ENBKP).

AEQB (Bit 08):	If set, this bit indicates that the contents of the Address Counter are equal to those of the Breakpoint Register. This bit is not affected by the state of ENBKP.
HALT (Bit 09):	This bit, when set, indicates that the MP is halted.
WAITOF (Bit 10):	When set, indicates that the MP is attempting to write to the output FIFO but the FIFO is full. This bit will clear as soon as space becomes available in the FIFO. The MP is in a "wait" condition during this time.
WAITIF (Bit 11):	When set, indicates that the MP is attempting to read from the input FIFO but the FIFO is empty. This bit will clear as soon as data becomes available in the FIFO. The MP is in a "wait" condition during this time.

6.3 Operation

There are two phases of operations for the Model 1240 microprocessor; Initialize and Execute.

6.3.1 Initialize

The first phase of operation is initializing the Control Store, the Quantizer tables, and the CSR. To initialize the Control Store the MP should first be reset. This insures that the MP is halted and that the control bits of the CSR are cleared. Next, the Address Counter is loaded with the address of the desired location within the Control Store in which a micro-instruction is to be stored. Three 16-bit words are next loaded into the specified location by writing into Control Store 1, 2, and 3 (refer to Table 6). The loaded microinstruction can be verified by reading Control Store 1, 2 and 3 and comparing with the original data. This process is repeated until the entire micro-program is loaded into Control Store.

The Quantizer Tables are initialized by the microprogram from Control Store. Data to be loaded into the Quantizer is sent to the input FIFO from the computer system via the interface. The microprogram moves data from the FIFO to the Data Register in one instruction cycle and then writes the output of the Data Register into the Quantizer during the next cycle (refer to Figure 36). The address for the Quantizer is supplied by the output of the MPU and is also controlled by the microprogram.

The control bits of the CSR are loaded by the computer to provide the desired operation of the MP. These bits control the modes of the FIFO's, the Address Counter's incrementing, the Breakpoint operation, the jump destination, and the STEP/RUN mode (refer to Section 6.2).

It will typically be desired to load the Address Counter and the Breakpoint Register with the microprogram starting address and the breakpoint address, respectively.

6.3.2 Execute

The execute phase is activated by writing into the MP addressable register for the START function (refer to Table 6). This removes the MP from the Halt state and begins execution of the microprogram. Execution starts at the microinstruction address in the Control Store specified by the Address Counter. This microinstruction is clocked into the Control Store Registers (refer to Figure 36) and the Address Counter is incremented for accessing of the next microinstruction.

During each execution cycle, data from one pipeline register is processed and/or moved to another pipeline register. The pipeline registers include the following (refer to Figure 36):

Data Register (D)	Scratch Pad Register (SP or S)
Multiplier Register (M)	Partial Product Register (PP or MI)
Quantizer Register (T)	FIFO Register (F)

The letters in parentheses are used to identify these registers in writing microinstructions.

Typical operation begins with inputting data from the input FIFO to the Data Register to be processed by the MPU. After processing, the data is moved to the Multiplier Register, the SP Register, or the T Register. If stored in the Multiplier Register, the data is multiplied by the microinstruction operand (X) and the result is clocked into the Partial Product Register. During the following instruction cycle this data passes through the adder and is then stored in the Data Register where it can be processed further by the MPU.

If the data from the MPU is placed in the Scratch Pad Register, it can then be stored in the 64-word Scratch Pad Memory where it can be accessed later. When accessed, this data can be sent to the Data Register for input to the MP or to the Multiplier Register for a multiply operation.

Data sent to the Quantizer Register is used to address the Quantizer Tables. The data at the location specified is then loaded into the Data Register and sent to the MPU, or it can be sent directly from the Quantizer to the FIFO Register.

The purpose of a pass through the MPU is to permit some processing on the way to the FIFO Register. The data in the FIFO Register is passed to the output FIFO during the next instruction cycle and then sent to the interface via MUXC.

In the RUN mode, the execution of data continues until the MP is Halted or a "wait" condition is encountered. In the STEP mode the MP, after receipt of a START command, executes one instruction cycle and then halts.

APPENDIX A

MICROPROGRAMMING WITH THE MODEL 1240 MICROASSEMBLER

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

INTRODUCTION

Data/Ware Development's D/W 1240 processor is a very high performance microprogrammed computer featuring a cycle time of about 150 nanoseconds. The machine is extensively overlapped and employs Schottky TTL LSI and MSI circuitry to achieve this high performance. The central building block is the Advanced Micro-Devices 4-bit microprocessor slice, the Am 2901. Three of these chips are combined to create a 12-bit word. The microinstructions are stored in a RAM memory, which can be loaded from a host computer. At present the PDP-11 is employed as the host, but any other computer could serve as well. Multiplication of two 12-bit numbers is carried out in a special asynchronous multiplier unit, which also has a 150 nsec cycle time.

Microinstructions consist of 48-bit words and thus can be loaded from the host machine as three 16-bit words. In the present model the microstore is a separate card which holds 1024 microinstructions. However, the backplane is wired so that up to four such cards can be plugged in if it is desired to write a longer program in microinstructions. Fig. A-1 shows the format of the microinstruction and Fig. 2 shows the overall logical organization.

Some of the significant features from Fig. A-2 are the CPU (Am 2901 chips), the multiplier (speeded up by forming two partial products and summing in an adder), the 64 word Scratch Pad Memory, and the input and output FIFO memories to simplify the problem of communicating with the host computer. The method of microprogramming the D/W 1240 using a cross-assembler written in FORTRAN is described in the text.

SECTION II

MICROINSTRUCTION FORMAT

The microinstruction format is as shown in Figure A-1 and the devices under microinstruction control are shown in Figure A-2. Bits 39-32, 31-28 and 24-16 control the AM2901 microprocessors. The bit assignments are given in Figure A-3, and the corresponding source language is described in Section 3.

Bit 47 is 1 whenever the Scratch Memory is being written. If bit 15 is 1, the Scratch Memory output is loaded into the M Register.

Bits 14-12 are used to form part of the Quantizer address. The contents of the T Register supply the rest.

Bits 0-11 are usually a 12-bit twos complement number by which the contents of the M Register are multiplied. The multiplication is performed in two stages. First two partial products are formed and held in the MI Register. Then on the next cycle, the partial products are added and their sum can be loaded into the D Register. Bits 0-11 can also be loaded into the D Register.

Register D can be loaded from any of seven sources, as indicated by bits 27-25. See Figure A-6 for the bit assignments.

The AM2901 microprocessor output can be directed to any of five destinations, as indicated by bits 18-16. Two bit patterns, however, are also used for other purposes. One causes the contents of the D Register to be written into the Quantizer. The other causes transfer of microcontrol. (Unless this bit pattern is present, microcontrol passes to the next microinstruction.) See Figure A-4 for details and bit assignments.

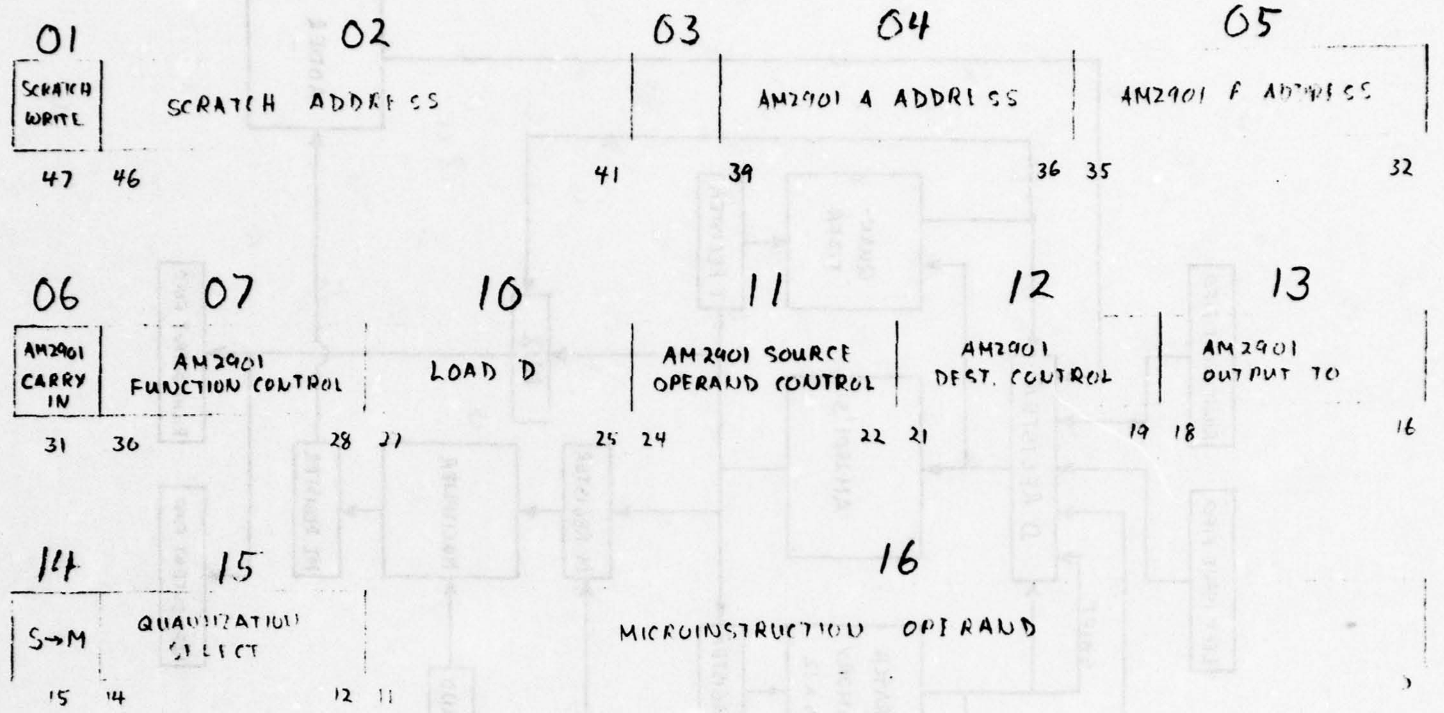


Figure A-1. Microinstruction bit assignments. The large octal numbers are field numbers.

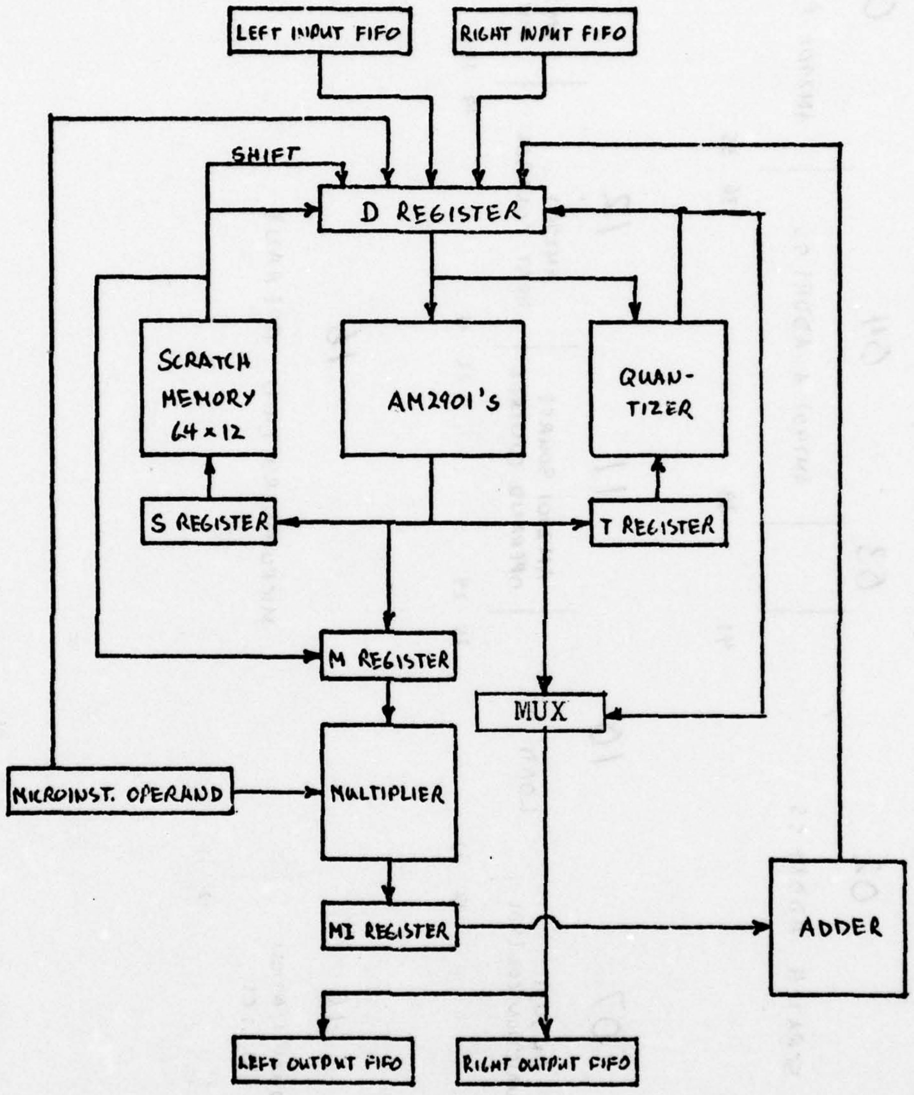
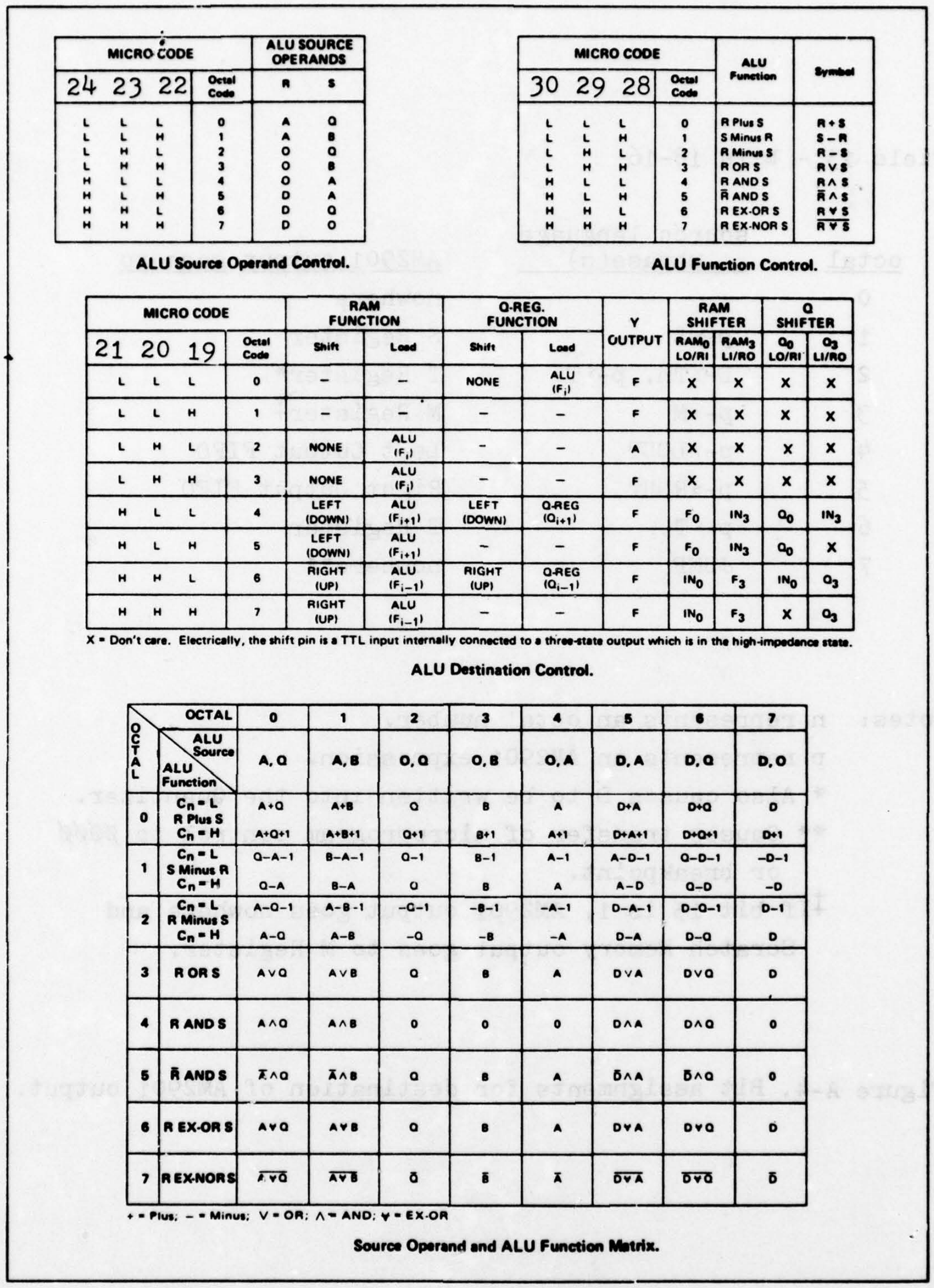


Figure A-2. Devices under Microinstruction Control



MICRO CODE				ALU SOURCE OPERANDS	
24	23	22	Octal Code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

ALU Source Operand Control.

MICRO CODE				ALU Function	Symbol
30	29	28	Octal Code		
L	L	L	0	R Plus S	$R+S$
L	L	H	1	S Minus R	$S-R$
L	H	L	2	R Minus S	$R-S$
L	H	H	3	R OR S	$R \vee S$
H	L	L	4	R AND S	$R \wedge S$
H	L	H	5	$\bar{R}$ AND S	$\bar{R} \wedge S$
H	H	L	6	R EX-OR S	$R \veebar S$
H	H	H	7	R EX-NOR S	$\overline{R \veebar S}$

ALU Function Control.

MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
21	20	19	Octal Code	Shift	Load	Shift	Load		RAM_0 LO/RI	RAM_3 LI/RO	Q_0 LO/RI	Q_3 LI/RO
L	L	L	0	–	–	NONE	ALU (F_i)	F	X	X	X	X
L	L	H	1	–	–	–	–	F	X	X	X	X
L	H	L	2	NONE	ALU (F_i)	–	–	A	X	X	X	X
L	H	H	3	NONE	ALU (F_i)	–	–	F	X	X	X	X
H	L	L	4	LEFT (DOWN)	ALU (F_{i+1})	LEFT (DOWN)	Q-REG (Q_{i+1})	F	F_0	IN_3	Q_0	IN_3
H	L	H	5	LEFT (DOWN)	ALU (F_{i+1})	–	–	F	F_0	IN_3	Q_0	X
H	H	L	6	RIGHT (UP)	ALU (F_{i-1})	RIGHT (UP)	Q-REG (Q_{i-1})	F	IN_0	F_3	IN_0	Q_3
H	H	H	7	RIGHT (UP)	ALU (F_{i-1})	–	–	F	IN_0	F_3	X	Q_3

X = Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state.

ALU Destination Control.

OCTAL	OCTAL	0	1	2	3	4	5	6	7
	ALU Function \ ALU Source	A, Q	A, B	0, Q	0, B	0, A	D, A	D, Q	D, 0
0	C_n = L R Plus S	$A+Q$	$A+B$	Q	B	A	$D+A$	$D+Q$	D
	C_n = H	$A+Q+1$	$A+B+1$	$Q+1$	$B+1$	$A+1$	$D+A+1$	$D+Q+1$	$D+1$
1	C_n = L S Minus R	$Q-A-1$	$B-A-1$	$Q-1$	$B-1$	$A-1$	$A-D-1$	$Q-D-1$	$-D-1$
	C_n = H	$Q-A$	$B-A$	Q	B	A	$A-D$	$Q-D$	$-D$
2	C_n = L R Minus S	$A-Q-1$	$A-B-1$	$-Q-1$	$-B-1$	$-A-1$	$D-A-1$	$D-Q-1$	$D-1$
	C_n = H	$A-Q$	$A-B$	$-Q$	$-B$	$-A$	$D-A$	$D-Q$	D
3	R OR S	$A \vee Q$	$A \vee B$	Q	B	A	$D \vee A$	$D \vee Q$	D
4	R AND S	$A \wedge Q$	$A \wedge B$	0	0	0	$D \wedge A$	$D \wedge Q$	0
5	$\bar{R}$ AND S	$\bar{A} \wedge Q$	$\bar{A} \wedge B$	Q	B	A	$\bar{D} \wedge A$	$\bar{D} \wedge Q$	0
6	R EX-OR S	$A \veebar Q$	$A \veebar B$	Q	B	A	$D \veebar A$	$D \veebar Q$	D
7	R EX-NOR S	$\overline{A \veebar Q}$	$\overline{A \veebar B}$	$\bar{Q}$	$\bar{B}$	$\bar{A}$	$\overline{D \veebar A}$	$\overline{D \veebar Q}$	$\bar{D}$

+ = Plus; – = Minus; $\vee$ = OR; $\wedge$ = AND; $\veebar$ = EX-OR

Source Operand and ALU Function Matrix.

Figure A-3. Bit assignments for AM2901 source, function and destination control fields.

Field 13 - bits 18-16

octal	source language phrase(s)	AM2901 output goes to
0		nowhere
1	p->S	S Register
2	D->Tn, p->T	T Register*
3	p->M	M Register‡
4	p->LOUT	Left Output FIFO
5	p->ROUT	Right Output FIFO
6	p->T	T Register
7	JUMP	nowhere**

Notes: n represents an octal number.

p represents an AM2901 expression.

* Also causes D to be written into the Quantizer.

** Causes transfer of microprogram control to ØØØØ or breakpoint.

‡If bit 15 is 1, AM2901 output goes nowhere and Scratch Memory output goes to M Register.

Figure A-4. Bit assignments for destination of AM2901 output.

SECTION III

SOURCE LANGUAGE FORMAT

Source language for the Model 1240 Microassembler is in ASCII format, and is generally taken from paper tape. Nulls, line feeds and rubouts are ignored. The carriage return is used as a line terminator. Form feeds are not permitted.

Each line of source language generates at most one microinstruction, and must not be any longer than 46 characters (not counting the carriage return that terminates it). Continuation lines are not permitted.

The source language on each line consists of items, which are logically grouped into phrases, and optional comments. A list of the permitted items is given in Figure A-5.

Almost any ASCII string enclosed in parentheses is treated as a comment. It will appear in the assembler listing but will not affect the assembly otherwise. Comments may be placed anywhere except inside items. A comment at the end of a line does not need a right parenthesis. Comments may not contain carriage returns or right parentheses. Nulls, rubouts and line feeds will not appear in the assembler listing.

Items are grouped logically into phrases. Items within a phrase must be in a particular order, but the phrases may appear in any order.

Items should be separated by one or more spaces, but spaces are not required before or after non-alphanumeric characters. Phrases may be separated by spaces and/or commas. (Spaces and commas are generally ignored except as item terminators.)

The item -> (formed with a minus sign and a right angle bracket) may be used interchangeably with -- or TO, or it may be omitted.

symbolic	meaning
Ø	Zero (as AM2901 operand only)
O	Zero (as AM2901 operand only)
D	D Register
Q	AM2901 Q Register
Rnn	AM2901 General Register nn
M	M Register
LIN	Left Input FIFO
RIN	Right Input FIFO
Tn	Quantizer entry
S	S Register
Snn	Scratch Memory location nn
nnnn	Microinstruction Operand is nnnn
LOUT	Left Output FIFO
ROUT	Right Output FIFO
T	T Register
JUMP	jump to ØØØØ or breakpoint
+	AM2901 addition
-	AM2901 subtraction
OR	AM2901 inclusive "or"
AND	AM2901 "and"
XOR	AM2901 exclusive "or"
CAND	AM2901 "mask" ($\bar{R}\wedge S$)
NXOR	AM2901 "exclusive nor" ($\overline{R\veebar S}$)
X	multiplied by
→	goes to
--	goes to
TO	goes to
SHIFTED	shifted (divided by 2)
ADDRESS	microcontrol store address
PATCH	patch microinstruction field
END	end of source language

Figure A-5. Permitted items in microassembler source language. Here "n", "nn" and "nnnn" mean a single octal digit, one or two octal digits, and four octal digits, respectively.

The following two-letter abbreviations may be substituted for the complete mnemonics:

abbreviation	complete mnemonic
LI	LIN
RI	RIN
LO	LOUT
RO	ROUT
JU	JUMP
AN	AND
XO	XOR
CA	CAND
NX	NXOR
SH	SHIFTED
AD	ADDRESS
PA	PATCH
EN	END

The phrases which load the D Register are shown in Figure 6. If no such phrase appears on a line, the default bit pattern Ø is used.

There are two kinds of phrases used to specify the AM2901 operation and the destination of the result:

```
    first form:  r op s -> dest
    second form:  -r+s -> dest
```

Here "op" means one of the AM2901 operations +, -, OR, AND, CAND, XOR or NXOR, "dest" means one of the destinations Rn, Q, Rn SHIFTED, S, T, M, LOUT or ROUT, and r and s are one of the permitted combinations.

r	s
Ø	Q
Ø	Rn
D	Ø
D	Q
D	Rn
Rn	Q
Rm	Rn

If the last combination is used and "dest" specifies a register, then it must be the same as that specified by s.

Other permitted phrases are as follows. In most cases, their meanings are clear.

phrase	meaning
S->Sn	Write S into location n of Scratch Memory.
D->Tn	Write D to Quantizer Memory.
Sn->M	Read location n in Scratch Memory to D.
nnnn X M	Multiply M by nnnn.
nnnn->D	Load nnnn into D.
JUMP	Jump to ØØØ or breakpoint.
PATCH nnnmm	Patch microinstruction field.

If the phrase D->Tn is used, then the output of the AM2901 is routed to T. Inconsistent phrases, such as RØ+R1->S, may not be used on the same line. A phrase such as RØ+R1->R1 is consistent with D->Tn, however, and will cause RØ+R1 to be loaded into both R1 and T.

The JUMP phrase is inconsistent with any that routes the AM2901 output anywhere outside the AM2901.

The PATCH nnnmm phrase, where nnnmm consists of from one to five octal digits, is used to generate bit patterns which cannot be generated in any other way. It puts the octal number nnn into field mm of the microinstruction, as shown in Figure A-1. Although it may not be possible to fill field 16 this way, it can be filled by nnnn X M. WARNING: Fields 06 and 07 are also used as control fields in some combinations. If these combinations are created by PATCH phrases, a control word will result.

The use of a PATCH phrase to fill a field also filled by a conventional phrase may produce an error flag for that field. However, MMMMMM NNNNNN PPPPPP (and the object tape) will contain the bit pattern generated by the PATCH phrase.

Field 10 - bits 27-25

octal	phrase	load D from-
0		(do not change D)
1	Sn->D	Scratch Memory output
2	Sn->D SHIFTED	Scratch Memory output ÷ 2
3	M->D	Adder output
4	LIN->D	Left Input FIFO
5	RIN->D	Right Input FIFO
6	Tn->D	Quantizer output
7	nnnn->D	Microinstruction Operand

Note: n represents one or two octal digits.
nnnn represents four octal digits.

Figure A-6. Phrases which load the D Register

There are also two assembler directives, which generate control words. Control words are not loaded into microcontrol store, but are used to modify the microcontrol store address or to end the microprogram. Only comments may appear on the same lines with assembler directives.

The assembler directive ADDRESS nnnn generates one control word with field 06 = 1, field 07 = 110_2 and field 16 = nnnn, where nnnn is a 4-digit octal number. It causes the next microinstruction to be loaded into location nnnn.

The assembler directive END generates one control word with field 06 = 1 and field 07 = 111_2. It signals the end of the source language.

Repetitious phrases on a single line are permitted, as long as they are consistent. For example, the phrases S67->D, S67->M, RØ-R1->Q, RØ-R1->T are perfectly acceptable on one line. However, S67->D, Ø137->D will be flagged as an error.

SECTION IV

LISTING FORMAT

When the microassembler is run in the listing mode, it produces one line of listing in the following format for each line of source code:

```
LLLL MMMMMM NNNNNN PPPPPP SSS---S
```

Here LLLL is the location of the microinstruction in microcontrol store. It is omitted if the line generated a control word. The listing fields MMMMMM, NNNNNN and PPPPPP are six-digit octal representations of bits 47-32, 31-16 and 15-Ø of the microinstruction (or control word), respectively. Notice that every field except fields 04 and 05 is represented by one or more octal digits in this format. The listing field SSS---S is a copy of the line of source language. If the line consisted entirely of comments, it will be the only listing field.

If errors are detected, a second line of error numbers is added. A complete list of possible errors is given in Figure A-7. An error in a particular microinstruction field is generally numbered to correspond to that field. Other errors are numbered as indicated. A single error may interfere with the scanner and generate many other errors. Since the microassembler is not designed primarily tc detect or correct errors, some errors may escape its notice.

error number	description
00	Field number out of range in PATCH
01	Scratch Write error
02	Scratch Address error
03	error in field 03
04	AM2901 A Address error
05	AM2901 B Address error
06	AM2901 Carry-In error
07	AM2901 Function error
10	error in data to be loaded into D
11	AM2901 Operand error
12	AM2901 Destination error
13	AM2901 Destination error
14	Sn->M error
15	Quantization Select error
16	Microinstruction operand error
17	too many items on one line (25 maximum)
20	unrecognized or misplaced character
21	unrecognized mnemonic
22	misplaced item

Figure A-7. Errors detected by the microassembler. Most error numbers correspond to the field numbers in Figure A-1.

SECTION V

OBJECT TAPE FORMAT

When the microassembler is run in the object mode, no listing is produced. An object tape is punched instead. The accompanying printout is not generally intelligible, since the object tape is in binary format.

Each microinstruction or control word is represented by six consecutive 8-bit words on the object tape. This is the most compressed form available, and with it microcode can be loaded at a rate of about 100 microinstructions per minute.

The last word in the object tape is a checksum.

Error messages are not punched into the object tape.

SECTION VI

OPERATING INSTRUCTIONS

To use the Model 1240 Microassembler, first load it in the usual way. The microassembler will type out its name (in abbreviated form) and wait.

To initiate the listing mode, type in the letter "L", put the source language tape into the reader, and turn on the reader.

To initiate the object mode, <u>first</u> turn on the paper tape punch, <u>then</u> type in the <u>letter</u> "O". The microassembler will punch a length of leader and pause. Then put in the source language tape and turn on the reader.

In either case, when the assembly is complete, the microassembler will again type out its name. Turn off the reader and/or punch and remove the source language and/or object tape. To perform another assembly, proceed again as stated above. To finish the assembly session, type in the letter "S".

To abort an assembly in either mode, simply turn off the reader and then type in (carriage return) END (carriage return).

APPENDIX B

DESCRIPTION OF SOFTWARE SUPPLIED WITH MODEL 1240 VIDEO PROCESSING SYSTEM

Prepared for
Wright-Patterson Air Force Base
Ohio, 45433
December 1, 1977

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

The 1240 operating system uses an ASCII teletype (or equivalent) to communicate with the operator. It can accept commands from the keyboard or from paper tape, if any, and does not distinguish between these two sources. When accepting commands from either source, it ignores nulls (blank tape), line feeds and rubouts, and does not use the parity bit.

In this manual, an overscore will be used to indicate a control character, i.e. $\overline{\text{P}}$ is the character generated by depressing both the control and "P" keys.

SECTION II

OPERATOR COMMANDS

A command is generally a combination of letters and octal numbers. The letters may, and the numbers must, be separated by one or more spaces. The octal numbers may be of any length, but only the rightmost digits are significant.

There are three kinds of commands:

1. "type" commands, which usually begin with the letter "T" and cause the system to type out specified registers or other information.
2. "change" commands, which begin with the letter "C" and cause the system to change the contents of specified registers, flags, etc.
3. "action" commands, which cause the microprocessor or system to take specified actions.

2.1 General Commands

2.1.1 Echo

Ordinarily, characters typed or read in will be echoed on the teletype printer. If $\overline{\text{F}}$ character is typed in, the echo will be inhibited, except that the bell will still ring. This feature is normally used when prepared sequences of commands are entered from paper tape and the echo is not wanted. The echo can be restored by typing or reading $\overline{\text{E}}$.

2.1.2 Ditto

If $\overline{D}$ ("ditto") is typed or read in, it will be interpreted as being the same as the character typed on the line directly above and will be echoed as such if the echo feature is enabled, as shown below.

operator types:	system echos:
CØ45 Ø Ø 123↵	CØ45 Ø Ø 123↵
$\overline{DD}5\overline{DDDDDDDD}4$↵	CØ55 Ø Ø 124↵

2.1.3 Abort

Each command normally occupies one line and is terminated by a carriage return. (A few exceptional commands are single control characters.) The system echos the command as it is entered, as long as the characters typed are part of a valid command. As soon as this condition is violated, the bell is rung and the offending character is <u>not</u> echoed. The system takes no other action until the carriage return is typed. The carriage return is not always echoed immediately--if the command produces output, it will normally be typed on the same line before the carriage return is echoed. If the command is recognized as invalid only after the carriage return is typed, a question mark is echoed. The command may be aborted by typing $\overline{X}$. There is no provision for deleting individual characters.

2.1.4 Comments

If a semicolon is typed, it and everything up to the next carriage return, $\overline{X}$ or $\overline{P}$ (whose function is described elsewhere) are echoed but are otherwise ignored. This feature is used to insert comments into the command listing.

2.2 Type Commands

2.2.1 TUn↵

Type the contents of UNIBUS location n, where n must be even.

2.2.2 TU$n_1 n_2$↵

Type the contents of UNIBUS locations n_1 to n_2, inclusive. The listing can be aborted by typing in a carriage return, except when the system is in the "paper tape" mode.

2.2.3 TU↵

Type the address and contents of UNIBUS location n+2, where n is the address of the last location typed.

2.2.4 M↵

Type the number of the microprocessor currently connected to the operating system.

2.2.5 Tn↵

Type the contents of control store location n in octal, partitioned into three 16-bit parts as in a microassembler listing.

2.2.6 T↵

Type the address and contents of control store location n+1, where n is the address of the last control store location typed.

2.2.7 Tr↵

Type register r, where r may be as follows:

r	meaning	comments
A	address register	
D	D register	type only
S	S register	type only
I	input FIFO	type last value deposited
J	output FIFO	type only
B	breakpoint	type last value deposited
P	control & status	
X	source	type last value deposited
Y	destination	type last value deposited

Where a register cannot be read, the value typed is the last value deposited by a Crn command, and is followed by an asterisk.

2.2.8 TIM↲ or TJM↲

Type the mode of the Input FIFO (P for "pixel", W for "word" as in the CxM commands), or the Output FIFO, respectively.

2.2.9 L↲

Type "D" or "E" to indicate whether the listing feature is disabled or enabled.

2.2.10 R↲

Type "F" or "S" to indicate whether the free-running or step mode is selected, respectively.

2.3 Change Commands

2.3.1 CUn_1n_2↲

Deposit n_2 into UNIBUS location n_1.

2.3.2 Mn↲

Connect the operating system to microprocessor number n, where n = 1 or 2. The following commands refer to the microprocessor which is connected in this way. The connection persists until changed.

2.3.3 Crn↲

Deposit n into register r where r is defined as in Section 2.2.7.

2.3.4 CxMP↲ or CxMW↲

Change mode of x to "pixel" (8-bit) or "word" (12-bit), respectively, where x is I for the Input FIFO and J for the Output FIFO.

2.3.5 C$n_1n_2n_3n_4$↲

Deposit $n_2n_3n_4$ into control store location n_1.

2.4 Action Commands

2.4.1 OSOPD↲

Disable the operating system overstore protection so the system can be patched with CUn_1n_2 commands.

2.4.2 OSOP↲

Restore the operating system overstore protection.

2.4.3 SNAP↲ or S↲

Take a picture and store it in Frame Store Memory.

2.4.4 PTB↲

Begin "paper tape" mode. This mode is used when commands such as TUn_1n_2 are read from paper tape.

2.4.5 PT↲

End "paper tape" mode.

2.4.6 EXIT↲

Exit from the operating system and halt. When the "continue" button is pressed, the Absolute Loader is actuated.

2.4.7 JSRn_1↲ or JSRn_1n_2↲ or JSR$n_1n_2n_3$↲

Jump to a user-coded subroutine whose entry address in n_1. The subroutine can return to the operating system by executing a RTS %7 instruction. When additional numbers are typed in, the subroutine can reference n_2 as (%5) and n_3 as 2(%5).

2.4.8 BD↲ or BE↲

Disable or enable the breakpoint feature, respectively.

2.4.9 JZ↲ or JB↲

Jump to zero or to the breakpoint, respectively.

2.4.10 LD↲ or LE↲

Disable or enable the listing feature, respectively.

2.4.11 RF↵ or RS↵

Select the free-running or step mode, respectively.

2.4.12 <control P>

Master clear the microprocessor.

2.4.13 <control G>

Start the microprocessor. If the step mode is selected, the microprocessor will halt after one step. If the free-running mode is selected, it will run until it is halted. If the listing feature is enabled, the address, microinstruction, and D register contents as altered by the microinstruction will be typed for each microinstruction. If the listing feature is enabled in the free-running mode, the listing may be aborted by typing a carriage return, except when the system is in the "paper tape" mode.

2.4.14 XQT$n_1n_2n_3$↵

Execute the microinstruction by loading it into control store, stepping the microprocessor, and then restoring the original contents of control store. The quantizer cannot be loaded in this manner, since quantizer loading instructions are pipelined in such a way that they cannot be executed singly.

2.4.15 LQT↵

Load the quantizer from core memory according to the following map:

	Core Location of Table Entry with -	
Table	Lowest (Negative) Address	Highest (Positive) Address
0	10000	10176
1	10200	10376
2	10400	10576
3	10600	10776
4	11000	11176
5	11200	11376
6	11400	11576
7	11600	12576

The table entry is stored in the six least significant bits of the indicated core location. Other core bits are ignored. When the operating system is loaded, this core area is filled as shown in Attachment 1. It may be altered with the CUn_1n_2 command, punched out with the Absolute Punch, and reloaded with the Absolute Loader.

<u>2.4.16 DUMP⤶</u>

Type out the contents of the address, D, S and Q registers, Am2901 internal registers R0 to R17, and Scratch Pad Memory locations S0 to S77, respectively. The registers and Scratch Pad Memory locations are restored after the dump. The listing cannot be aborted.

<u>2.4.17 LMPn⤶</u>

Load a microassembler object tape from the high-speed reader into control store, incrementing all addresses by n. If n is omitted, it is taken to be zero. The operating system will type out the bottom and top addresses of the segments loaded.

2.5 Messages to the Operator

2.5.1

The message "BAD ADDRESS" means that an attempt has been made to reference an odd or nonexistent UNIBUS location or a nonexistent microprocessor or register, or to write into a read-only register or memory location.

2.5.2

The message "RUNNING" means that an attempt has been made to read the S or D register of a microprocessor or change its configuration while it is running. When run status is requested, however, this is not an error message, but merely indicates that the microprocessor is running.

TABLE B-1

COMMAND INDEX

Section	Command (N)	Types out	Notes
2.2.1	TUn	UNIBUS location n	(U)
2.2.2	TU$n_1 n_2$	UNIBUS locations n_1 to n_2	(U),(A)
2.2.3	TU	UNIBUS location n+2	(X),(U)
2.2.4	M	number of current micro-processor	
2.2.5	Tn	microinstruction at location n	(I)(H)
2.2.6	T	microinstruction at location n+1	(X),(I) (H)
2.2.7	TD	microprocessor D register	(H)
2.2.7	TS	microprocessor S register	(H)
2.2.7	TP	microprocessor control and status	
2.2.7	TI	last word pushed into input FIFO	(*)
2.2.7	TJ	word popped from output FIFO	(P)
2.2.7	TB	microprocessor breakpoint	(*)
2.2.7	TX	source control	(*)
2.2.7	TY	destination control	(*)
2.2.7	TA	address of current micro-instruction	(H)(E)
2.2.8	TIM	input FIFO mode (P or W)	
2.2.8	TJM	output FIFO mode (P or W)	
2.2.9	L	listing status of current m microprocessor (E or D)	
2.2.10	R	run status of current microprocessor (F or S)	

TABLE B-1

COMMAND INDEX

(continued)

Section	Command(N)	Changes	To	Notes
2.3.1	CUn_1n_2	UNIBUS location n_1	n_2	(U)
2.3.2	Mn	current micro-processor	n(1 or 2)	
2.3.3	Crn			
2.3.3	CIn	input FIFO mode	n	(Q),(H)
2.3.3	CPn	microprocessor control & status	n	(H)
2.3.3	CBn	microprocessor breakpoint	n	(H)
2.3.3	CXn	source control	n	(H)
2.3.3	CYn	destination control	n	(H)
2.3.3	CAn	current micro-instruction address	n	(H),(E)
2.3.4	CIMP	input FIFO mode	pixel	(H)
2.3.4	CIMW	input FIFO mode	word	(H)
2.3.4	CJMP	output FIFO mode	pixel	(H)
2.3.4	CJMW	output FIFO mode	word	(H)
2.3.5	C$n_1n_2n_3n_4$	microinstruction at location n_1	$n_2n_3n_4$	(H),(I)

TABLE B-1

COMMAND INDEX

(continued)

Section	Command(N)	Does		Notes
2.4.1	OSOPD	overstore protection	disabled	
2.4.2	OSOP	overstore protection	enabled	
2.4.3	SNAP	take a picture		
2.4.4	PT	begin paper tape commands		
2.4.5	PTE	end paper tape commands		
2.4.6	EXIT	exit from operating system		
2.4.7	JSR	jump to user coded subroutine		
2.4.8	BD	breakpoint	disabled	(H)
2.4.8	BE	breakpoint	enabled	(H)
2.4.9	JZ	jump status	zero	(H)
2.4.9	JB	jump status	breakpoint	(H)
2.4.10	LE	listing status	listing	(H)
2.4.10	LD	listing status	no listing	(H)
2.4.11	RF	run status	free	(H)
2.4.11	RS	run status	single step	(H)
2.4.12	$\overline{P}$	master clears microprocessor		(C)
2.4.13	$\overline{G}$	starts or steps microprocessor		(C)
2.4.14	XQT$n_1n_2n_3$	executes micro-instruction $n_1n_2n_3$		(I),(H)
2.4.15	LQT	load quantizer tables		(H)
2.4.16	DUMP	dump microprocessor contents		(H),(D)
2.4.17	LMP	load microprogram from paper tape		(H)

Notes to Command Table

(N) n, n_1, n_2, n_3, n_4 indicate octal numbers, separated by spaces if necessary

(U) The UNIBUS locations must be even and represent whole words. Special registers usually have UNIBUS locations, but they should not be handled by this command because of possible interference with system error protection features.

(A) The listing produced by this command may be aborted by typing $\overline{X}$.

(H) Microprocessor must be halted.

(*) The word typed out is not actually read from the hardware register in which it resides. It is a copy of what was last written out, and is followed by an asterisk.

(P) The output FIFO is popped when this command is executed. If it is empty, an appropriate message will appear.

(I) The microinstruction format is AAAA BBBB CCCC, where AAAA, BBBB and CCCC represent bits 47-32, 31-16 and 15-0, respectively.

(X) Here n is the address last typed.

(Q) Pushes n into input FIFO. If the FIFO is already full, an appropriate message will appear.

(C) A one-character command not necessarily followed by a carriage return.

(D) Types out microprocessor D, S and Q registers, general registers RØ-R17 and scratch memory registers SØ-S77, in that order.

(E) The "current" microinstruction is the first one to be executed when the microprocessor is started.

SECTION III

OPERATING PROCEDURES

3.1 Applying Power

Apply power as indicated in Table B-2.

TABLE B-2

POWER-UP SEQUENCE

Step	Action	Location	Indication
1	Switch on Camera	Rear of Camera Housing	Red Lamp will Light
2	Switch on TTY	TTY	
3	Connect System to AC power	Rear of Rack	
4	Switch on "AC MAIN"	AC Panel (119510)	Amber Lamp will Light
5	Switch on "ADC"	DC Panel (119520)	Red Lamp will Light
6	Switch on "TVI"	DC Panel (119520)	Red Lamp will Light
7	Switch on "FSM"	DC Panel (119520)	Red Lamp will Light
8	Switch on "MP"	DC Panel (119520)	Red Lamp will Light
9	Switch on Expansion Box	Expansion Box	Red Lamp will Light
10	Switch CPU to "DC ON"	CPU	"DC ON" will Light
11	Switch on H.S. Tape Reader	H.S. Tape Reader	
12	Switch on Monitor	Inside Door of Monitor	

Note: Reverse order of steps when turning system off.

3.2 Start-Up Procedure

3.2.1 Loading the Absolute Loader

Boot the CPU from the front panel. Put the ABSOLUTE LOADER -HS PATCH tape into the teletype reader (not the high-speed reader). Notice that the leader portion of this tape does not consist of nulls. Turn on the reader and type in TT2.

The teletype reader will read to the blank area in the middle of the tape and halt. Press the "continue" button on the front panel. The Teletype reader will read to the blank area at the end of tape, but it will not halt. As soon as it reaches the blank area at the end of tape, halt the CPU and make the following patches:

```
      deposit  000167  into location  77620
      deposit  177454  into location  77622
```

3.2.2 Loading the Operating System

Put the Operating System tape into the high-speed reader and start the CPU at location 077500 (the Absolute Loader). The computer should read the entire tape and then halt. If it halts before the end of tape, or if it reads to the end of tape but does not halt, load the Operating System tape again. Other segments can be loaded in the same way.

Start the Operating System at location 000000. It will respond by typing out its name. Both microprocessors will be master cleared.

The Operating System can be manually restarted at location 000000 or at location 005074. When restarted at 000000, it will master clear both microprocessors, but options previously set will not be affected. When restarted at 005074, it will be ready to accept commands, but nothing will be done to the microprocessors.

The Operating System is then prepared to accept commands from the Teletype keyboard.

3.3 Writing Routines to be Used With the Operating System

User-coded routines can be called by the Operating System by means of the JSR command. Return to the operating system can be accomplished by an RTS %7 or a TRAP 20 instruction.

A user-coded routine can access many registers and flags in the microprocessors and Operating System as shown in Table B-3. The TRAP instruction is used by the Operating System to access many of its own subroutines. Those which can be used to good advantage by user-coded subroutines are described in detail in the remainder of this section.

TRAP Ø

Read or accept a character from the teletype reader or keyboard into the less significant byte of RØ and clear the other byte. Do not process special characters. Usually used for reading binary tapes.

TRAP 2

Type, or type and punch, the character in the less significant byte of RØ. Usually used for punching binary tapes.

TRAP 4

Read or accept a character from the teletype reader or keyboard into bits 7-Ø of RØ. Clear the parity bit. Ignore nulls, line feeds and rubouts. Process the special characters <control D>, <control E> and <control F>.

TRAP 6

Type the characters in bits 7-Ø of RØ. Inhibit output if this option is in effect. Supply a line feed after each carriage return.

TRAP 1Ø

```
    .BYTE c1,a1-.
    .BYTE c2,a2-.
         :
    .BYTE cn,an-.
    .BYTE Ø,a-.
```

TABLE B-3
REGISTERS AND FLAGS

Action	MP1	MP2	Current MP
master clear (w)	CLR 164000	CLR 164040	CLR @ 1000
start (w)	CLR 164020	CLR 164060	CLR @ 1020
halt (w)	CLR 164002	CLR 164042	CLR @ 1022
D register (R)	164000	164040	@ 1000
S register (R)	164004	164044	@ 1004
Source register (w)	164012	164052	@ 1012
Dest. register (w)	164010	164050	@ 1016
Input FIFO (w)	164016	164056	@ 1016
Output FIFO (R)	164016	164056	@ 1016
Control & Status	164022	164062	@ 1022
Breakpoint (w)	164026	164066	@ 1026
Microinst. address	164030	164070	@ 1030
Microinst. bits 15-0	164032	164072	@ 1032
Microinst. bits 31-16	164034	164074	@ 1034
Microinst. bits 47-32	164036	164076	@ 1036
run option (P)	1074	1114	@ 1054
list option (Q)	1076	1116	@ 1056

(R) = read only
(w) = write only
(P) = ASCII "F" or "S"
(Q) = ASCII "D" or "E"

If (RØ) = c_k, branch to a_k; otherwise branch to a. Note c_1 = Ø is permitted. Usually used to branch on characters, since bit 7 of RØ must be zero. Note a_k-. must be nonnegative; only forward branches are allowed.

```
TRAP 12
```
Type a carriage return and line feed.

```
TRAP 14
+ pointer
+n
```
Type a space and then n octal digits of data, where "pointer" is the address of the address of the data.

```
TRAP 16
.ASCII /message/
.BYTE Ø
.EVEN
```
Type the message, which is any string of ASCII characters not containing a null.

```
JSR %5, 64Ø6
```
Read a character from the high speed reader into the less significant byte of RØ and clear the other byte.

```
JSR %5, 5726
.WORD (bits 47-32)
.WORD (bits 31-16)
.WORD (bits 15-0)
```

Load the microinstruction from the calling sequence into location 0 of the current microprocessor, execute it, and then restore location 0.

3.4 Image Compression

3.4.1 Operation

1. Load the Operating System in the manner set forth in Section 3.2.2.
2. Load the forward and inverse transforms into microprocessors 1 and 2, respectively, with LMP commands.
3. Load the Quantizer in microprocessor 1 with the LQT instruction.
4. Load the forward and inverse transform patches for the desired compression mode into microprocessors 1 and 2, respectively, with LMP commands (otherwise no compression will occur).
5. Type JSR 134ØØ↲ to start the microprocessors. Type a second carriage return to stop the microprocessors and return to the Operating System.
6. To change to a different mode of compression, start over again at step 4.

3.4.2 Forward DCT

3.4.2.1 Method

The pixels $g_0, g_1, \ldots, g_{31}$ come in through the Input FIFO's in that order. Pixels with even subscripts ($g_0, g_2, g_4 \ldots, g_{30}$) come in through the right FIFO and other pixels come in through the left FIFO. Therefore, the Input FIFO must be in the pixel mode.

The (scaled) DCT is computed:

$$A_k = \sum_{j=0}^{31} g_j \cos\left[(j+\tfrac{1}{2})k\theta\right]$$

$$k = 0, 1, \ldots, 31$$

$$\theta = \pi/32$$

According to the Data/Ware algorithm,

$$A_k = R_e(w^{\frac{1}{2}k}B_k), \quad k = 0,1,\ldots,16$$

$$A_{32-k} = -\mathrm{Im}(w^{\frac{1}{2}k}B_k), \quad k = 1,2,\ldots,15$$

$$w = e^{i\theta}$$

where $B_0, B_1, B_2, \ldots, B_{31}$ are the DFT (discrete Fourier transform) of the pixels in a different order, viz., $g_0, g_2, g_4, \ldots, g_{30}, g_{31}, g_{29}, \ldots, g_1$. The verification of this fact is straightforward.

The coefficients B_k are calculated by a modification of the Fast Fourier transform which takes advantage of the fact that the input data are real. This method computes only the coefficients B_0 through B_{16}. The other coefficients are their conjugates, and are not used in the DCT calculation in any event. The DFT coefficients used in the calculations are listed in Figure B-1.

<u>3.4.2.2 Calculations</u>

The coefficients $I_0, I_1, I_2, J_0, J_1, \ldots, P_1, P_2$ of DFT's of length 4 are first computed directly from the definition. The DFT coefficients of a,b,c,d are (a+c)+(b+d), (a-c)+(d-b)i, (a+c)-(b+d), respectively. The results are stored as indicated in column (A) of Figure 2. The real parts of $I_1, J_1, \ldots, P_1$, are stored in RØ-R17 and S2Ø-S37, the imaginary parts are stored in SØ-S17. The negatives of J_2, N_2, L_2 and P_2 are stored for convenience in later processing, instead of J_2, N_2, L_2 and P_2 themselves.

The DFT's of length 8 are computed from the DFT's of length 4 by means of the FFT algorithm, modified to take advantage of the symmetries created by real data (see Section 3.4.6). The results are stored as indicated in column (B) of Figure B-2. Real parts of complex coefficients are generally stored in RØ-R17 and S2Ø-S37, imaginary parts are stored in SØ-S17. The exceptions are noted. Only 45° butterflies are used in this step.

The DFT's of length 16 are then computed from the DFT's of length 8 by the "next column of butterflies", and the results are stored as indicated in column (C) of Figure 2.

Then $B_0, B_1, \ldots, B_{16}$ are computed by the "last column of butterflies", and are stored as indicated in column (D) of Figure 2.

Finally, $A_{16}, A_0, A_1, A_{31}, A_2, A_{30}, A_3, A_{29}, A_4, A_{28}, A_5, A_{27}, A_6, A_{26}, A_7, A_{25}, A_8, A_{24}, A_9, A_{23}, A_{10}, A_{22}, A_{11}, A_{21}, A_{12}, A_{20}, A_{13}, A_{19}, A_{14}, A_{18}, A_{15}, A_{17}$ are computed, in that order. As each DCT coefficient is computed, a DPCM is performed on it and the result is sent to the Output FIFO in 12-bit form, except for A_{28}, A_{29}, A_{30} and A_{31}, which are computed but not sent to the Output FIFO. Therefore, the Output FIFO must be in the word mode.

3.4.2.3 Forward DCT-Scaling

Each pixel g_j is in the range $0 \leq g_j \leq 63$. Hence

$$0 \leq A_0 \leq \sum_{j=0}^{31} 63 = 1953$$

and for $k \neq 0$

$$\begin{aligned} A_k &= \sum_{j=0}^{31} (g_j - 31.5) \cos\left[(j+\tfrac{1}{2})k\theta\right] \\ &\quad + 31.5 \sum_{j=0}^{31} \cos\left[(j+\tfrac{1}{2})k\theta\right] \\ &= \sum_{j=0}^{31} (g_j - 31.5) \cos\left[(j+\tfrac{1}{2})k\theta\right] \\ |A_k| &\leq \sum_{j=0}^{31} 31.5 = 976.5 \end{aligned}$$

The intermediate DFT coefficients are within the same or tighter bounds, so there is no overflow with 12-bit arithmetic, which can handle integers from -2048 to +2047, inclusive.

3.4.3 Forward DPCM

3.4.3.1 Method

The DPCM is illustrated in Figure B-3. The value of α currently used is 0.95. Scratch memory location $4\emptyset_8$+k is used to hold the stored value of A_k. Quantization is assigned as indicated in Figure 4 when the system is run in the data compression mode. When not in this mode, the system uses Quantizer table 3 for all coefficients. The contents of the quantization tables used are shown in Figure B-5.

3.4.3.2 Scaling

To prove that overflow will not occur in the DPCM calculations is a little more difficult. If x represents the contents of the scratch storage location corresponding to A_k, the DPCM calculations are

$$f(x) = Q(A_k - \alpha x) + \alpha x \rightarrow x$$

where Q(z) is the output of the quantization table when the input is z.

The initial value of x is quite arbitrary, so $A_k - \alpha x$ may overflow until the DPCM has settled. The values of x could conceivably oscillate in such a way that $A_k - \alpha x$ always overflows, but this is very unlikely. (It can be prevented by initializing x to zero.) For coefficients other than A_0, the attenuation factor will eventually bring x into the range $-1071 \leq x \leq 1071$. It must now be shown that x will remain there, and hence $A_k - \alpha x$ cannot overflow.

Since Q is a nondecreasing function, the maximum value of f(x) is attained when $A_k = 976$, its maximum value. In this case

$$f(x) = Q(976 - \alpha x) + \alpha x$$

$$= \left[Q(976 - \alpha x) - (976 - \alpha x)\right] + 976$$

the maximum value of the expression in brackets is 79, and it is attained when x = 1071 and Table 0 is used. Hence $f(x) \leq 1055$.

Similarly, the minimum value of f(x) is attained when $A_k = -976$, and

$$f(x) = \left[Q(-976-\alpha x)-(-976-\alpha x)\right] -976$$

The minimum value is attained when Table 0 is used and x = -1071
Hence

$$f(x) \geq -1055$$

The coefficiecnt A_0 is special. In this case x must be in the range $-94 \leq x \leq 2048$ in order that $A_0-\alpha x$ will not overflow. As before f(x) is maximized when

$$f(x) = \left[Q(1953-\alpha x)-(1953-\alpha x)\right] +1953$$

which occurs when x = 2048 and Table 0 is used, and hence $f(x) \leq 2032$. Similarly, f(x) is minimized when

$$f(x) = Q(-\alpha x)-(-\alpha x)$$

which occurs when x = -94 and Table 0 is used, and hence $f(x) \geq -78$.

3.4.4 Inverse DPCM

The inverse DPCM involves only calculations already performed for the forward DPCM, so there is no overflow. The inverse DPCM reconstructs $A_0, A_1, \ldots, A_{27}$ and stores them in Scratch Memory locations S40 through S77, respectively.

3.4.5 Inverse DCT

The inverse DCT is computed by reversing all the computations done for the forward DCT. However, some scaling is done to keep the results from overflowing. The basic butterfly derived in Section 3.4.6,

$$A_0 = B_0+C_0,\ A_N = B_0-C_0$$

$$A_{\frac{1}{2}N} = B_{\frac{1}{2}N}-i\ C_{\frac{1}{2}N}$$

$$A_k = B_k+w^{-k}C_k$$

$$A_{N-k} = \overline{B_k-w^{-k}C_k}$$

can easily be reversed:

$$B_0 = \tfrac{1}{2}(A_0+A_N),\ C_0 = \tfrac{1}{2}(A_0-A_N)$$

$$B_{\frac{1}{2}N} = \mathrm{Re}A_{\frac{1}{2}N},\ C_{\frac{1}{2}N} = -\mathrm{Im}A_{\frac{1}{2}N}$$

$$B_k = \tfrac{1}{2}(A_k+\overline{A_{N-k}})$$

$$C_k = \tfrac{1}{2}w^k(A_k-\overline{A_{N-k}})$$

Since division by 2 is time-consuming, the inverse butterflies compute $2B_0$, $2C_0$, $2B_{\frac{1}{2}N}$, $2C_{\frac{1}{2}N}$, $2B_k$ and $2C_k$. In the inverse final rotations, all sines and cosines are divided by 4, so that $\frac{1}{4}B_0$, $\frac{1}{4}B_1$, ..., $\frac{1}{4}B_{16}$ are computed from the stored values of A_0, A_1,..., A_{27}, with A_{28}, A_{29}, A_{30} and A_{31} taken to be zero.

Figure 2 shows how the scaled reconstructed coefficients are stored. Column (E) represents the storage after the inverse final rotations. Columns (F), (G) and (H) represent the results of the three groups of butterflies.

The last butterflies are computed from formulas of the form

$$8g_0 = 2I_0+2I_2+4\mathrm{Re}I_1$$

$$8g_{31} = 2I_0+2I_2-4\mathrm{Re}I_1$$

<u>These symbols</u>	<u>represent the low-frequency coefficients of the DFT of:</u>
B_0-B_{16}	$g_0, g_2, g_4, \ldots, g_{30}, g_{31}, g_{29}, g_{27}, \ldots, g_1$
C_0-C_8	$g_0, g_4, g_8, \ldots, g_{28}, g_{31}, g_{27}, g_{23}, \ldots, g_3$
D_0-D_8	$g_2, g_6, g_{10}, \ldots, g_{30}, g_{29}, g_{25}, g_{21}, \ldots, g_1$
E_0-E_4	$g_0, g_8, g_{16}, g_{24}, g_{31}, g_{23}, g_{15}, g_7$
F_0-F_4	$g_4, g_{12}, g_{20}, g_{28}, g_{27}, g_{19}, g_{11}, g_3$
G_0-G_4	$g_2, g_{10}, g_{18}, g_{26}, g_{29}, g_{21}, g_{13}, g_5$
H_0-H_4	$g_6, g_{14}, g_{22}, g_{30}, g_{25}, g_{17}, g_9, g_1$
I_0, I_1, I_2	$g_0, g_{16}, g_{31}, g_{15}$
J_0, J_1, J_2	g_8, g_{24}, g_{23}, g_7
K_0, K_1, K_2	$g_4, g_{20}, g_{27}, g_{11}$
L_0, L_1, L_2	$g_{12}, g_{28}, g_{19}, g_3$
M_0, M_1, M_2	$g_2, g_{18}, g_{29}, g_{13}$
N_0, N_1, N_2	$g_{10}, g_{26}, g_{21}, g_5$
O_0, O_1, O_2	g_6, g_{22}, g_{25}, g_9
P_0, P_1, P_2	$g_{14}, g_{30}, g_{17}, g_1$

A_0-A_{31} are the DCT of $g_0, g_1, g_2, \ldots, g_{31}$

Figure B-1. DFT Coefficients Used in DCT Calculations

	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
R0(S20)	I_1	E_1	C_1	(B_1)	$\frac{1}{4}B_0$	$\frac{1}{2}C_0$	E_0	$2I_0$
R1(S21)	P_0	H_0	D_0	(B_0)	$\frac{1}{4}B_1$	$\frac{1}{2}C_1$	E_1	$2I_1$
R2(S22)	M_1	G_1	D_1	(B_{15})	$\frac{1}{4}B_2$	$\frac{1}{2}C_2$	E_2	$2J_0$
R3(S23)	L_0	F_0	C_0	(B_{16})	$\frac{1}{4}B_3$	$\frac{1}{2}C_3$	E_3	$2J_1$
R4(S24)	K_1	(F_1)	(D_6)	(B_7)	$\frac{1}{4}B_4$	$\frac{1}{2}C_4$	F_0	$2L_0$
R5(S25)	N_0	G_0	$(-D_8)$	(ImB_8)	$\frac{1}{4}B_5$	$\frac{1}{2}C_5$	F_3	$2L_1$
R6(S26)	O_1	(H_1)	(D_7)	(B_9)	$\frac{1}{4}B_6$	$\frac{1}{2}C_6$	F_2	$2K_0$
R7(S27)	J_0	E_0	C_4	(B_4)	$\frac{1}{4}B_7$	$\frac{1}{2}C_7$	F_1	$2K_1$
R10(S30)	(J_1)	E_3	C_3	(B_3)	$\frac{1}{4}B_8$	$\frac{1}{2}D_0$	H_0	$2O_0$
R11(S31)	O_0	$(-H_4)$	$C_7(ImD_4)$	(B_{10})	$\frac{1}{4}B_9$	$\frac{1}{2}D_7$	H_1	$2O_1$
R12(S32)	(N_1)	G_3	(D_3)	(B_{13})	$\frac{1}{4}B_{10}$	$\frac{1}{2}D_6$	H_2	$2P_0$
R13(S33)	K_0	$(-F_4)$	$C_6(ImC_4)$	(B_6)	$\frac{1}{4}B_{11}$	$\frac{1}{2}D_5$	H_3	$2P_1$
R14(S34)	(L_1)	(F_3)	C_5	(B_5)	$\frac{1}{4}B_{12}$	$\frac{1}{2}D_4$	G_0	$2N_0$
R15(S35)	M_0	G_2	(D_2)	(B_{14})	$\frac{1}{4}B_{13}$	$\frac{1}{2}D_3$	G_3	$2N_1$
R16(S36)	(P_1)	(H_3)	(D_5)	(B_{11})	$\frac{1}{4}B_{14}$	$\frac{1}{2}D_2$	G_2	$2M_0$
R17(S37)	I_0	E_2	C_2	(B_2)	$\frac{1}{4}B_{15}$	$\frac{1}{2}D_1$	G_1	$2M_1$
S0	I_2	E_4	C_8	ReB_8	$\frac{1}{4}B_{16}$	$\frac{1}{2}C_8$	E_4	$2I_2$
S1	P_1	H_3	D_5	B_{11}	$\frac{1}{4}B_1$	$\frac{1}{2}C_1$	E_1	$2I_1$
S2	M_2	G_4	ReD_4	ReB_{12}	$\frac{1}{4}B_2$	$\frac{1}{2}C_2$	E_2	$2J_2$
S3	L_1	F_3	C_5	B_5	$\frac{1}{4}B_3$	$\frac{1}{2}C_3$	E_3	$2J_1$
S4	K_2	ReF_2		B_4	$\frac{1}{4}B_4$	$\frac{1}{2}C_4$	F_4	$2L_2$
S5	N_1	G_3	D_3	B_{13}	$\frac{1}{4}B_5$	$\frac{1}{2}C_5$	F_3	$2L_1$
S6	O_2	ReH_2		B_{12}	$\frac{1}{4}B_6$	$\frac{1}{2}C_6$	F_2	$2K_2$
S7	J_1	E_3	C_3	B_3	$\frac{1}{4}B_7$	$\frac{1}{2}C_7$	F_1	$2K_1$
S10	$-J_2$	E_2	C_2	B_2	$\frac{1}{4}B_8$	$\frac{1}{2}D_8$	H_4	$2O_2$
S11	O_1	H_1	D_7	B_9	$\frac{1}{4}B_9$	$\frac{1}{2}D_7$	H_1	$2O_1$
S12	$-N_2$	G_2	D_2	B_{14}	$\frac{1}{4}B_{10}$	$\frac{1}{2}D_6$	H_2	$2P_2$
S13	K_1	F_1	C_7	B_7	$\frac{1}{4}B_{11}$	$\frac{1}{2}D_5$	H_3	$2P_1$
S14	$-L_2$	F_2	C_6	B_6	$\frac{1}{4}B_{12}$	$\frac{1}{2}D_4$	G_4	$2N_2$
S15	M_1	G_1	D_1	B_{15}	$\frac{1}{4}B_{13}$	$\frac{1}{2}D_3$	G_3	$2N_1$
S16	$-P_2$	H_2	D_6	B_{10}	$\frac{1}{4}B_{14}$	$\frac{1}{2}D_2$	G_2	$2M_2$
S17	I_1	E_1	C_1	B_1	$\frac{1}{4}B_{15}$	$\frac{1}{2}D_1$	G_1	$2M_1$

Figure B-2. Storage Layout at Various Stages of Processing

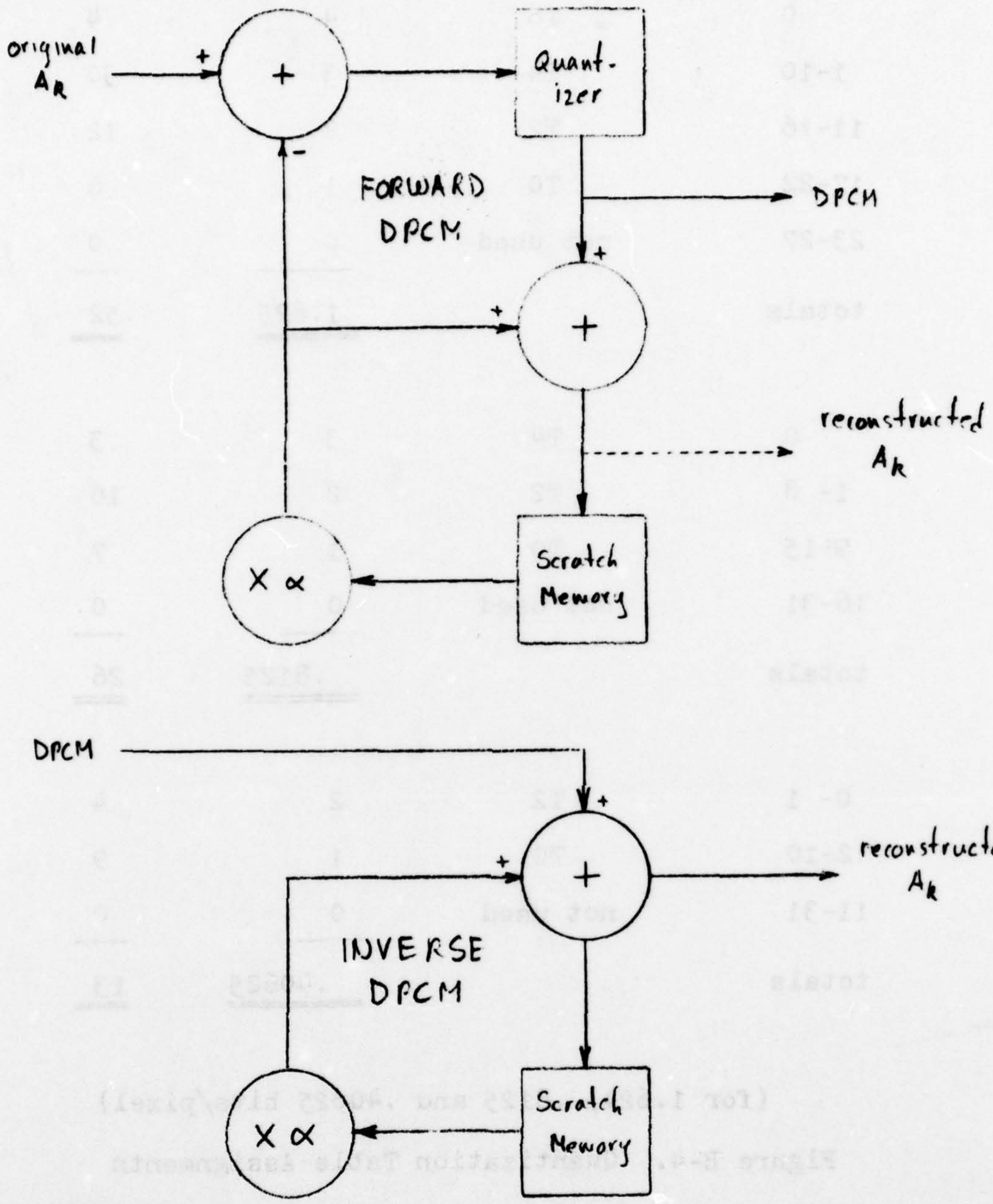


Figure B-3. Forward and Inverse DPCM's

DCT Frequencies	Table	Bits Each	Total Bits
0	T6	4	4
1-10	T4	3	30
11-16	T2	2	12
17-22	T0	1	6
23-27	not used	0	0
totals		1.625	52
0	T4	3	3
1- 8	T2	2	16
9-15	T0	1	7
16-31	not used	0	0
totals		.8125	26
0- 1	T2	2	4
2-10	T0	1	9
11-31	not used	0	0
totals		.40625	13

(for 1.624, .8125 and .40625 bits/pixel)

Figure B-4. Quantization Table Assignments

Table 6

input from	input to	output
-2048	-191	-232
-192	-137	-160
-136	-97	-112
-96	-65	-80
-64	-49	-56
-48	-33	-40
-32	-17	-24
-16	-1	-8
0	15	8
16	31	24
32	47	40
48	63	56
64	95	80
96	135	112
136	191	160
192	2047	232

Table 4

input from	input to	output
-2048	-137	-192
-136	-65	-96
-64	-33	-48
-32	-1	-16
0	31	16
32	63	48
64	135	96
136	2047	192

Table 2

input from	input to	output
-2048	-33	-68
-32	-1	-16
0	31	16
32	2047	68

Table 0

input from	input to	output
-2048	-1	-16
0	2047	16

Figure B-5. Contents of Quantization Tables which come directly from the definitions. The scaled pixels $8g_0, 8g_1, \ldots, 8g_{31}$ are then sent to the Output FIFO in the 8-bit mode, in which bits 10-3 form the 8-bit pixels $g_0, g_1, \ldots, g_{31}$. If a small roundoff error produces a negative pixel value, it is forced to zero by the output logic.

3.4.6 Modified FFT for Real Data

Let

$$A_k = \sum_{j=0}^{2N-1} a_j w^{-jk}, \quad k=0,1,\ldots,N$$

$$w = \exp(\pi i/N)$$

where $a_0, a_1, \ldots, a_{2N-1}$ are real. Notice that only some of the FFT coefficients are used——$A_{N+1}, A_{N+2}, \ldots, A_{2N-1}$ are not used in this method.

The usual FFT decomposition is $A_k = B_k + w^{-k} C_k$, where

$$B_k = \sum_{j=0}^{N-1} a_{2j} w^{-2jk}$$

$$C_k = \sum_{j=0}^{N-1} a_{2j+1} w^{-2jk}$$

are DFT's of length N with real data. Notice that since $w^{2N}=1$

$$\overline{B}_{N-k} = \sum_{j=0}^{N-1} a_{2j} w^{-2jk} = \overline{B}_k$$

Similarly, $\overline{C}_{N-k} = C_k$ Hence

$$\overline{A}_{N-k} = \overline{B}_{N-k} + w^{N-k} \overline{C}_{N-k}$$

$$= B_k - w^{-k} C_k$$

since $w^N = -1$. The modified FFT butterflies are

$$A_k = B_k + w^{-k} C_k$$

$$A_{N-k} = \overline{B_k - w^{-k} C_k}$$

for k=1,2,...,$\frac{1}{2}$N-1. The special cases k=0, k=$\frac{1}{2}$N and k=N reduce to

$$A_0 = B_0 + C_0, \quad A_N = B_0 - C_0$$

$$A_{\frac{1}{2}N} = B_{\frac{1}{2}N} - iC_{\frac{1}{2}N}$$

It is easily shown that $A_0, B_0, C_0, B_{\frac{1}{2}N}, C_{\frac{1}{2}N}$ and A_N are real.

3.4.7 Changing Quantization and Compression

The quantization tables assigned to various DPCM coefficients, and hence the data rate compression, may be changed by patching bits 14-12 (the digit X in NNNNNN NNNNNN NXNNNN) of the following microinstructions in microprocessor 1:

DCT Frequency	Address	DCT Frequency	Address
0	362	14	571
1	367	15	603
2	401	16	356
3	413	17	610
4	425	18	576
5	437	19	564
6	451	20	552
7	463	21	540
8	475	22	526
9	507	23	514
10	521	24	502
11	533	25	470
12	545	26	456
13	557	27	444

The inverse DPCM code in microprocessor 2 should be patched to force unused DCT coefficients to zero. If this is done, then the quantization tables used to compute these coefficients in the forward transform are, of course, irrelevant. Inverse DPCM code for the DPCM coefficient corresponding to the nth DCT frequency is as follows:

```
a.  0mmXXX  XXXXXX  1XXXXX   Sm--M
b.  XXXXXX  XX5XXX  XX3632   3632 x M  RIN--D
c.  XXXXXX  003700  XXXXXX   M--D  D+Ø--Q
d.  XXXXXX  00X611  XXXXXX   D+Q--S
e.  1mmXXX  XXXXXX  XXXXXX   S--Sm
```

where $m=40_8+n$ and the X's represent irrelevant code or code for other frequencies. The appropriate patch is to line (d) as follows:

```
XXXXXX  00X611  XXXXXX   D+Q--S  (ORIGINAL)
XXXXXX  04X711  XXXXXX   D and O--S  (PATCHED)
```

The locations to be so patched are as follows:

DCT frequency	location	DCT frequency	location
0	005	14	063
1	007	15	067
2	011	16	003
3	013	17	071
4	015	18	065
5	017	19	061
6	023	20	055
7	027	21	051
8	033	22	045
9	037	23	041
10	043	24	035
11	047	25	031
12	053	26	025
13	057	27	021

3.4.8 Roundoff Errors

The pixels at the left edge of each stripe are especially sensitive to roundoff errors, especially at high data rates. The picture is much improved by removing these pixels and extending adjacent pixels to fill the space, which can be done by modifying only three microinstructions in locations 530, 531 and 532 of microprocessor number 2 as shown in Figure 6. The tape supplied with the system contains this modification. The original code can be restored by loading the tape labeled "first pixel unfix" into microprocessor number 2 _after_ loading the full inverse DPCM and DCT tape. The modification can then be reintroduced, if desired, by loading the tape labeled "first pixel fix".

3.5 Miscellaneous Routines

3.5.1 Control Store Punch

To punch the leader and initialize the checksum, type JSR 14ØØØ, turn on the punch, and type a carriage return. When the punch stops, turn it off and type another carriage return.

To punch locations n_1 to n_2 in Control Store, inclusive, type JSR 141ØØ n_1n_2, turn on the punch, and type a carriage return. When the punch stops, turn it off and type another carriage return. To punch other segments of Control Store, repeat the steps in this paragraph as many times as necessary.

To punch a stop code, checksum and trailer, type JSR 143ØØ, turn on the punch, and type a carriage return. When the punch stops, turn it off and type another carriage return.

3.5.2 Absolute Loader

Put the tape to be loaded into the high-speed reader. Start the Absolute Loader at 077500. If the computer stops _before_ the end of the tape, or if it does not stop _at_ the end of the tape, the load should be repeated. A JSR 77500 command can be used to enter the Absolute Loader, but the operating system must be manually restarted at 000000 or 005074 after the load is finished.

```
ORIGINAL CODE

                                   (SO→D ABOVE)
0527 000000 000500 000000  D+RO→Q
0530 000020 000615 000000  R1+Q → ROUT
0531 012020 111011 000000  -R1+Q→S   S12→D
0532 013240 111500 000000            -D+R12→Q  S13→D
0533 137000 000614 000000   S→S37  D+Q→LOUT
```

```
FIRST PIXEL  FIXED = SECOND PIXEL

                                   (SO→D ABOVE)
0527 000000 000500 000000  D+RO→Q
0530 012020 111011 000000  -R1+Q→S   S12→D
0531 013240 111500 000000            -D+R12→Q  S13→D
0532 137000 000615 000000   S→S37  D+Q→ROUT
0533 137000 000614 000000   S→S37  D+Q→LOUT
```

Figure B-6. Modification of Pixels at Left Edges of Stripes

3.5.3 Absolute Punch

The Absolute Punch can be used to punch out on tape the contents of any contiguous segment of core memory, such as the Quantizer image. The tape can be loaded with the Absolute Loader.

To use the Absolute Punch, first use the CU command to load the lower and upper limits of the core area to be punched into locations 074212 and 074214, respectively. Then type in JSR 74ØØØ, turn on the punch, and type a carriage return. When the computer halts, restart the system at 0 or 5074.

The Absolute Punch produces no trailer. Therefore, the area punched should extend slightly beyond the area of interest so nothing will be lost when the tape is torn off.

Alternatively, the command JSR 153ØØ <bottom><top> can be used to activate the Absolute Punch.

3.5.4 Special Load and Punch

A special absolute load and punch routine can be used to punch out and reload core areas other than the operating system or special load and punch routine. The routine is most useful for punching and reloading Quantizer images.

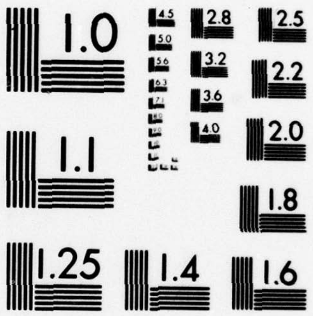
To punch a tape of a core area, type the command JSR 73ØØØ <bottom> <top> turn on the punch, and type a carriage return. The tape will have a trailer of reasonable length, so the exact top address can be given. When the punch stops, turn it off and type a carriage return.

To load a tape punched by the special load and punch routine, put it into the high speed reader and type in the command JSR 732ØØ2 . The routine will type out the bottom and top addresses of the area loaded, followed by an asterisk if there is a checksum error.

Tapes produced by the regular and special absolute punch routines are not compatible.

AD-A072 917 DATA/WARE DEVELOPMENT INC SAN DIEGO CALIF F/G 9/2
ADVANCED DIGITAL TV SYSTEM.(U)
FEB 79 P J ERDELSKY, R V KEELE, G G MURRAY F33615-77-C-1198
UNCLASSIFIED AFAL-TR-79-1006 NL

3 OF 4
AD
A072917


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

3.5.5 Demonstration Routine

Initialize operating system as described in Operating System procedure. Place demonstration object tape in the H.S. tape reader. Next place the demonstration tape in the teletype reader and start the reader. The tapes will read in and a description of the system operation will be typed on the teletype while the results are displayed on the monitor.

The Demonstration routine has not been re-written since initial delivery and must be run with the original Operating System.

3.5.6 Demonstration Programs

Two simple programs, occupying locations 017000-017526 of core memory, perform simple transformations for test and demonstration purposes. *All processing is done by the PDP-11, not by the microprocessors.* The programs are included on the Operating System tape or may be loaded individually from the tape labeled "DEMO/DEMI".

The command JSR 17ØØ2 calls the "posterizing" routine. This routine takes a picture and then sets the three least significant bits of each pixel to zero, a process which makes many images resemble silk screen posters. When the entire image has been processed, the routine takes another picture and starts over again. Delays are inserted so the eye can follow the process, which continues until the Operating System is manually restarted at location 0 or 5074.

The command JSR 173ØØ2 calls the "negative" routine. This routine takes a picture and then subtracts each pixel from 63_{10}, a process which makes the picture appear to be a photographic negative. When the entire image has been processed, the routine takes another picture and starts over again. Delays are inserted so the eye can follow the process, which continues until the Operating System is manually restarted at location 0 or 5074.

ATTACHMENT 1 TO APPENDIX B

MEMORY MAP

core area	segment	entry point(s)
000000-007776	Operating System	000000 005074
010000-012576	Quantizer Image	
013000-013244	Interrupt Handler	013000
014000-014420	Control Store Punch	014000 014200 014300
017000-017226	Posterizing Demonstrator	017000
017300-017526	Complementing Demonstrator	017300
074000-074214	Absolute Punch	074000
077200-077776	Absolute Loader	077500

ATTACHMENT 2 TO APPENDIX B

Software Listings

PRECEDING PAGE NOT FILMED
BLANK

```
                         (ASSEMBLED 10-5-77)
                         (MICROCODE FOR DCT AND DPCM)
                         (5 OCT 77)

                         (LOAD FIRST 16 PIXELS)

0000 000000 000200 000000 0+Q->Q (NOP)
0001 000000 005611 000000 RIN->D,  D+Q->S
0002 161000 004730 000000 D+0->R0,  LIN->D, S->S61
0003 000001 005730 000000 D+0->R1,  RIN->D
0004 000002 004730 000000 D+0->R2,  LIN->D
0005 000003 005730 000000 D+0->R3,  RIN->D
0006 000004 004730 000000 D+0->R4,  LIN->D
0007 000005 005730 000000 D+0->R5,  RIN->D
0010 000006 004730 000000 D+0->R6,  LIN->D
0011 000007 005730 000000 D+0->R7,  RIN->D
0012 000010 004730 000000 D+0->R10, LIN->D
0013 000011 005730 000000 D+0->R11, RIN->D
0014 000012 004730 000000 D+0->R12, LIN->D
0015 000013 005730 000000 D+0->R13, RIN->D
0016 000014 004730 000000 D+0->R14, LIN->D
0017 000015 005730 000000 D+0->R15, RIN->D
0020 000016 004730 000000 D+0->R16, LIN->D
0021 000017 005730 000000 D+0->R17, RIN->D


                         (LOAD NEXT 16 AND FORM SUM AND DIFF)

0022 000360 110511 000000 -D+R17->S
0023 117377 004530 000000  D+R17->R17   S->S17, LIN->D
0024 000340 000511 000000  D+R16->S
0025 116340 115511 000000 -D+R16->S,   S->S16, RIN->D
0026 136320 110511 000000 -D+R15->S,   S->S36
0027 115335 004530 000000  D+R15->R15   S->S15, LIN->D
0030 000300 000511 000000  D+R14->S
0031 114300 115511 000000 -D+R14->S   S->S14, RIN->D
0032 134260 110511 000000 -D+R13->S   S->S34
0033 113273 004530 000000  D+R13->R13   S->S13, LIN->D
0034 000240 000511 000000  D+R12->S
0035 112240 115511 000000 -D+R12->S   S->S12, RIN->D
0036 132220 110511 000000 -D+R11->S   S->S32
0037 111231 004530 000000  D+R11->R11   S->S11, LIN->D
0040 000200 000511 000000  D+R10->S
0041 110200 115511 000000 -D+R10->S   S->S10, RIN->D
0042 130160 110511 000000 -D+R7->S   S->S30
0043 107167 004530 000000  D+R7->R7   S->S7, LIN->D
0044 000140 000511 000000  D+R6->S
0045 106146 115530 000000 -D+R6->R6, S->S6, RIN->D
0046 000120 110511 000000 -D+R5->S
0047 105125 004530 000000  D+R5->R5, S->S5, LIN->D
0050 000100 000511 000000  D+R4->S
0051 104104 115530 000000 -D+R4->R4, S->S4, RIN->D
0052 000060 110511 000000 -D+R3->S
0053 103063 004530 000000  D+R3->R3, S->S3, LIN->D
```

```
0054 000040 000511 000000  D+R2->S
0055 102042 115530 000000 -D+R2->R2, S->S2, RIN->D
0056 000020 110511 000000 -D+R1->S
0057 101021 004530 000000  D+R1->R1, S->S1, LIN->D
0060 000000 000500 000000  D+R0->Q
0061 000000 110530 000000 -D+R0->R0

                          (MORE SUMS AND DIFFERENCES)

0062 000360 110011 000000 -R17+D->S
0063 010377 001530 000000  R17+D->R17  S10->D
0064 100160 110511 000000 S->S0         -D+R7->S
0065 004167 001530 000000               D+R7->R7  S4->D
0066 110260 120511 000000               S->S10    D-R13->S
0067 014273 001530 000000                         D+R13->R13  S14->D
0070 104060 110511 000000                         S->S4       -D+R3->S
0071 002063 001530 000000 S2->D                               D+R3->R3
0072 114320 120511 000000 D-R15->S                            S->S14
0073 012335 001530 000000 D+R15->R15   S12->D
0074 102120 110511 000000 S->S2         -D+R5->S
0075 006125 001530 000000               D+R5->R5  S6->D
0076 112220 110511 000000               S->S12    -D+R11->S
0077 016231 001530 000000                         D+R11->R11    S16->D
0100 106020 110511 000000                         S->S6       -D+R1->S
0101 000021 001530 000000 S0->D                               D+R1->R1
0102 116367 120111 000000 R17-R7->S                           S->S16

                          (COLUMN OF 45 DEGREE BUTTERFLIES)

                          (S0->D ABOVE)
                          (R17-R7->S ABOVE)
0103 100367 000130 000000 R17+R7->R7, S->S0
0104 000017 000730 000000 D+0->R17
0105 000263 110111 000000 -R13+R3->S
0106 133263 000130 000000 R13+R3->R3, S->S33
0107 002525 121111 000000                         R15-R5->S, S2->D
0110 102525 000130 000000 (*********)             R15+R5->R5, S->S2
0111 007015 000730 100000 S7->M       (**********) D+0->R15
0112 030221 110111 102650 2650 X M    S30->M   (*)-R11+R1->S
0113 131221 003130 002650 M->D         2650 X M(*)R11+R1->R1, S->S31
0114 000000 003700 000000 D+0->Q      M->D     (***************
0115 000000 000600 000000 D+Q->Q
0116 007010 120030 102650 R0 -Q->R10 2650 X M    S7->M
0117 000000 003030 002650 R0 +Q->R0  M->D        2650 X M
0120 000000 003700 000000            D+0->Q      M->D
0121 017000 121600 000000                        D-Q->Q    S17->D
                          (*********
0122 003000 000611 100000 S3->M      (************) D+Q->S
0123 034000 000010 102650 2650 X M    S34->M   (*)
0124 117000 113611 002650 M->D         2650 X M(*)S->S17   -D+Q->S
0125 107000 003700 000000 D+0->Q      M->D     (*)          S->S7
0126 000000 000600 000000 D+Q->Q               (****************
0127 003100 120011 102650 R4 -Q->S    2650 X M    S3->M
0130 134100 003011 002650 R4 +Q->S    M->D        2650 X M S->S34
```

```
0131 124000 003700 000000             D+0->Q     M->D     S->S24
0132 015000 121600 000000                        D-Q->Q   S15->D
                          (*********
0133 005000 000611 100000 S5->M   (************) D+0->S
0134 032000 000010 102650 2650 X M    S32->M  (*)
0135 113000 113611 002650 M->D        2650 X M(*)S->S13   -D+0->S
0136 103000 003700 000000 D+0->Q      M->D    (*)         S->S3
0137 000000 000600 000000 D+Q->Q              (***************
0140 005052 120030 102650 R2 -Q->R12 2650 X M    S5->M
0141 000042 003030 002650 R2 +Q->R2  M->D        2650 X M
0142 000000 003700 000000            D+0->Q      M->D
0143 015000 121600 000000                        D-Q->Q   S15->D
                          (*********
0144 001000 000611 100000 S1->M   (************) D+0->S
0145 036000 000010 102650 2650 X M    S36->M  (*)
0146 115000 113611 002650 M->D        2650 X M(*)S->S15   -D+0->S
0147 105000 003700 000000 D+0->Q      M->D    (*)         S->S5
0150 000000 000600 000000 D+Q->Q              (***************
0151 001140 120011 102650 R6 -Q->S    2650 X M   S1->M
0152 136140 003011 002650 R6 +Q->S    M->D       2650 X M S->S36
0153 126000 003700 000000             D+0->Q     M->D     S->S26
0154 011000 121600 000000                        D-Q->Q   S11->D
                          (*********
0155 000000 000611 000000         (************) D+Q->S
0156 111000 110611 000000                     (*)S->S11   -D+0->S
0157 101000 000010 000000                     (*)         S->S1
                                              (*******************


                          (MICROCODE FOR DCT AND DPCM - TAPE 2)
                          (19 AUG 77)

                          (NEXT COLUMN OF BUTTERFLIES)

0160 000163 121111 000000                        R7-R3->S, S0->D
0161 100163 000130 000000 (***********)          R7+R3->R3, S->S0
0162 013007 000730 100000 S13->M     (***********)D+0->R7
0163 024121 110111 101420 1420 X M    S24->M  (*)-R5+R1->S
0164 125121 003130 003544 M->D        3544 X M(*)R5+R1->R1, S->S25
0165 000000 003700 000000 D+0->Q      M->D    (*******************)
0166 000000 000600 000000 D+Q->Q
0167 013011 120030 101420 R0 -Q->R11 1420 X M    S13->M
0170 000000 003030 003544 R0 +Q->R0  M->D        3544 X M
0171 000000 003700 000000            D+0->Q      M->D
0172 017000 121600 000000                        D-Q->Q   S17->D
                          (**********)
0173 014000 000611 100000 S14->M    (***********)D+0->S
0174 004000 000010 102650 2650 X M    S4->M   (*)
0175 117000 113611 002650 M->D        2650 X M(*)S->S17   -D+0->S
0176 113000 003700 000000 D+0->Q      M->D    (*)         S->S13
0177 000000 000600 000000 D+Q->Q              (*************
0200 014373 120030 102650 R17-Q->R13 2650 X M    S14->M
0201 000377 003030 002650 R17+Q->R17 M->D        2650 X M
0202 000000 003700 000000            D+0->Q      M->D
```

```
0203 010000 121600 000000                       D-Q->Q   S10->D
                          (**********)
0204 003000 000611 100000 S3->M      (***********)D+Q->S
0205 034000 000010 103544 3544 X M  S34->M  (*)
0206 110000 113611 001420 M->D       1420 X M(*)S->S10   -D+Q->S
0207 114000 003700 000000 D+Q->Q     M->D    (*)        S->S14
0210 000000 000600 000000 D+Q->Q             (*************
0211 003214 120030 103544 R10-Q->R14 3544 X M   S3->M
0212 000210 003030 001420 R10+Q->R10 M->D       1420 X M
0213 000000 003700 000000            D+Q->Q     M->D
0214 007000 121600 000000                       D-Q->Q   S7->D
                          (**********)
0215 011000 000611 100000 S11->M     (***********)D+Q->S
0216 026000 000010 101420 1420 X M  S26->M  (*)
0217 107000 113611 003544 M->D       3544 X M(*)S->S7    -D+Q->S
0220 103000 003700 000000 D+Q->Q     M->D    (*)        S->S3
0221 000000 000600 000000 D+Q->Q             (*************
0222 011040 120011 101420 R2 -Q->S   1420 X M   S11->M
0223 126042 003030 003544 R2 +Q->R2  M->D       3544 X M S->S26
0224 000000 003700 000000            D+Q->Q     M->D
0225 015000 121600 000000                       D-Q->Q   S15->D
                          (**********)
0226 016000 000611 100000 S16->M     (***********)D+Q->S
0227 006000 000010 102650 2650 X M  S6->M   (*)
0230 115000 113611 002650 M->D       2650 X M(*)S->S15   -D+Q->S
0231 111000 003700 000000 D+Q->Q     M->D    (*)        S->S11
0232 000000 000600 000000 D+Q->Q             (*************
0233 016320 120011 102650 R15-Q->S   2650 X M   S16->M
0234 124320 003011 002650 R15+Q->S   M->D       2650 X M S->S24
0235 135000 003700 000000            D+Q->Q     M->D     S->S35
0236 012000 121600 000000                       D-Q->Q   S12->D
                          (**********)
0237 001000 000611 100000 S1->M      (***********)D+Q->S
0240 036000 000010 103544 3544 X M  S36->M  (*)
0241 112000 113611 001420 M->D       1420 X M(*)S->S12   -D+Q->S
0242 116000 003700 000000 D+Q->Q     M->D    (*)        S->S16
0243 000000 000600 000000 D+Q->Q             (*************
0244 001240 120011 103544 R12-Q->S   3544 X M   S1->M
0245 136240 003011 001420 R12+Q->S   M->D       1420 X M S->S36
0246 132000 003700 000000            D+Q->Q     M->D     S->S32
0247 005000 121600 000000                       D-Q->Q   S5->D
                          (**********)
0250 015000 000611 100000 S15->M     (***********)D+Q->S
0251 105000 113611 000000                    (*)S->S5    -D+Q->S
0252 101061 120111 000000            R3-R1->S(*)        S->S1
                                             (****************


                          (LAST COLUMN OF BUTTERFLIES)

                          (S15->M ABOVE)
                          (R3-R1->S ABOVE)
0253 123040 000413 000620 0620 X M  S+R2->M             S->S23
```

```
0254 000061 003111 003731 M->D       3731 X M   R3+R1->S
0255 121000 003700 000000 D+Q->Q     M->D                S->S21
0256 000000 000600 000000 D+Q->Q
0257 015000 120011 100620 R8 -Q->S   0620 X M   S15->M
0260 122000 003011 [illegible] R8 +Q->S   M->D       [illegible] X M S->S22
0261 120000 003700 000000            D+Q->Q     M->D      S->S20
0262 017000 121600 000000                       D-Q->Q    S17->D
                          (***********)
0263 012000 000611 100000 S12->M   (***********)D+Q->S
0264 035000 000010 101420 1420 X M   S35->M  (*)
0265 117000 113611 003544 M->D       3544 X M(*)S->S17    -D+Q->S
0266 115000 003700 000000 D+Q->Q     M->D    (*)          S->S15
0267 000000 000600 000000 D+Q->Q             (*************
0270 012360 120011 101420 R17-Q->S   1420 X M   S12->M
0271 135360 003011 003544 R17+Q->S   M->D       3544 X M S->S35
0272 137000 003700 000000            D+Q->Q     M->D      S->S37
0273 010000 121600 000000                       D-Q->Q    S10->D
                          (***********)
0274 005000 000611 100000 S5->M    (***********)D+Q->S
0275 032000 000010 102162 2162 X M   S32->M  (*)
0276 110000 113611 003247 M->D       3247 X M(*)S->S10    -D+Q->S
0277 112000 003700 000000 D+Q->Q     M->D    (*)          S->S12
0300 000000 000600 000000 D+Q->Q             (*************
0301 005200 120011 102162 R10-Q->S   2162 X M   S5->M
0302 132200 003011 003247 R10+Q->S   M->D       3247 X M S->S32
0303 130000 003700 000000            D+Q->Q     M->D      S->S30
0304 007000 121600 000000                       D-Q->Q    S7->D
                          (***********)
0305 031000 000611 100000 S31->M   (***********)D+Q->S
0306 002000 000010 102650 2650 X M   S2->M   (*)
0307 107000 113611 002650 M->D       2650 X M(*)S->S7     -D+Q->S
0310 105000 003700 000000 D+Q->Q     M->D    (*)          S->S5
0311 000000 000600 000000 D+Q->Q             (*************
0312 031160 120011 102650 R7 -Q->S   2650 X M   S31->M
0313 102160 003011 002650 R7 +Q->S   M->D       2650 X M S->S2
0314 127000 003700 000000            D+Q->Q     M->D      S->S27
0315 033000 121600 000000                       D-Q->Q    S33->D
                          (***********)
0316 001000 000611 100000 S1->M    (***********)D+Q->S
0317 036000 000010 103247 3247 X M   S36->M  (*)
0320 104000 113611 002162 M->D       2162 X M(*)S->S4     -D+Q->S
0321 106000 003700 000000 D+Q->Q     M->D    (*)          S->S6
0322 000000 000600 000000 D+Q->Q             (*************
0323 001300 120011 103247 R14-Q->S   3247 X M   S1->M
0324 136300 003011 002162 R14+Q->S   M->D       2162 X M S->S36
0325 134000 003700 000000            D+Q->Q     M->D      S->S34
0326 003000 121600 000000                       D-Q->Q    S3->D
                          (***********)
0327 016000 000611 100000 S16->M   (***********)D+Q->S
0330 024000 000010 103544 3544 X M   S24->M  (*)
0331 103000 113611 001420 M->D       1420 X M(*)S->S3     -D+Q->S
0332 101000 003700 000000 D+Q->Q     M->D    (*)          S->S1
0333 000000 000600 000000 D+Q->Q             (*************
0334 016260 120011 103544 R15-Q->S   3544 X M   S16->M
```

```
0335 151260 003011 001420 R13+Q->S   M->D       1420 X M S->S31
0336 133000 003700 000000            D+Q->Q     M->D     S->S33
0337 014000 121600 000000                       D-Q->Q   S14->D
                          (**********)
0340 011000 000611 100000 S11->M   (***********)D+Q->S
0341 026000 000010 103731 3731 X M   S26->M  (*)
0342 114000 113611 000620 M->D       0620 X M(*)S->S14   -D+Q->S
0343 116000 003700 000000 D+Q->Q     M->D    (*)         S->S16
0344 000000 000600 000000 D+Q->Q             (************
0345 011220 120011 103731 R11-Q->S   3731 X M   S11->M
0346 176220 003011 000620 R11+Q->S   M->D       0620 X M S->S26
0347 124000 003700 000000            D+Q->Q     M->D     S->S24
0350 013000 121600 000000                       D-Q->Q   S13->.
                          (**********)
0351 060000 000611 100000 S60->M   (***********)D+Q->S
0352 023000 000010 103632 3632 X M   S23->M  (*)
0353 113000 113611 002650 M->D       2650 X M(*)S->S13   -D+Q->S
0354 111000 003700 000000 D+Q->Q     M->D    (*)         S->S11
                                             (****************


                          (MICROCODE FOR DCT AND DPCM - TAPE 3)
                          (5 OCT 77)

                          (FINAL ROTATIONS AND DPCM)

                          (S60->M  ABOVE)
                          (3632 X M, S23->M ABOVE)
                          (M->D, 2650 X M ABOVE)
                          (D+Q->Q, M->D ABOVE)
0355 040000 120616 100000 D-Q->T                         S40->M
0356 000000 006010 033632            T3->D (A16)         3632 X M
                          (*********)
0357 020000 003611 100000 S20->M  (*)D+Q->S              M->D
0360 021000 001700 000000         (*)                    D+Q->Q S21->D
0361 160000 120616 000000         (*)S->S60              D-Q->T
                                  (************)
0362 017000 006010 133775 3775 X M   S17->M  (***********) T3->D (A0)
0363 041000 003611 100144 M->D       0144 X M  S41->M  (*) D+Q->S
0364 140000 003700 003632 D+Q->Q     M->D      3632 X M(*) S->S40
0365 000000 003600 000000          D+Q->Q   M->D         (************
0366 020000 110616 100000          -D+Q->T  S20->M
0367 017000 006700 130144 D+Q->Q   T3->D (A1) 0144 X M   S17->M
0370 077000 003611 103775          D+Q->S   M->D   3775 X M, S77->M
0371 141000 003700 003632          S->S41   D+Q->Q    M->D, 3632 X M
0372 000000 113600 000000 (*************************)-D+Q->Q, M->D
0373 037000 110616 100000 S37->M                        (*)-D+Q->T
0374 010000 000700 103766 3766 X M S10->M          (*)D+Q->Q, (T3->D)(A01)
0375 042000 003611 100310 M->D      0310 X M  S42->M  (*)D+Q->S
0376 177000 003700 003632 D+Q->Q   M->D      3632 X M(*)S->S77
                                                  (***************
0377 000000 003600 000000          D+Q->Q   M->D
0400 037000 110616 100000          -D+Q->T  S37->M
```

```
0401 010000 006700 130310 D+C->Q   T3->D (A2) 0310 X M   S10->M
0402 076000 003611 103766          D+C->S   M->D   3766 X M, S76->M
0403 142000 003700 003632          S->S42   D+C->Q   M->D, 3632 X M
0404 000000 113600 000000 (***************************)-D+C->Q, M->D
0405 030000 110616 100000 S30->M                        (*)-D+C->T
0406 007000 000700 103751 3751 X M S7 ->M        (*)D+C->Q, (T3->D)(A30)
0407 043000 003611 100454 M->D      0454 X M S43->M  (*)D+C->S
0410 176000 003700 003632 D+C->Q   M->D      3632 X M(*)S->S76
                                                      (***************
0411 000000 003600 000000          D+C->Q   M->D
0412 030000 110616 100000          -D+C->T  S30->M
0413 007000 006700 130454 D+C->Q   T3->D (A3) 0454 X M   S7->M
0414 075000 003611 103751          D+C->S   M->D   3751 X M, S75->M
0415 143000 003700 003632          S->S43   D+C->Q   M->D, 3632 X M
0416 000000 113600 000000 (***************************)-D+C->Q, M->D
0417 027000 110616 100000 S27->M                        (*)-D+C->T
0420 004000 000700 103730 3730 X M S4->M         (*)D+C->Q, (T3->D)(A29)
0421 044000 003611 100617 M->D      0617 X M S44->M  (*)D+C->S
0422 175000 003700 003632 D+C->Q   M->D      3632 X M(*)S->S75
                                                      (***************
0423 000000 003600 000000          D+C->Q   M->D
0424 027000 110616 100000          -D+C->T  S27->M
0425 004000 006700 130617 D+C->Q   T3->D(A4) 0617 X M   S4->M
0426 074000 003611 103730          D+C->S   M->D   3730 X M, S74->M
0427 144000 003700 003632          S->S44   D+C->Q   M->D, 3632 X M
0430 000000 113600 000000 (***************************)-D+C->Q, M->D
0431 034000 110616 100000 S34->M                        (*)-D+C->T
0432 003000 000700 103702 3702 X M S3->M         (*)D+C->Q, (T3->D)(A28)
0433 045000 003611 100761 M->D      0761 X M S45->M  (*)D+C->S
0434 174000 003700 003632 D+C->Q   M->D      3632 X M(*)S->S74
                                                      (***************
0435 000000 003600 000000          D+C->Q   M->D
0436 034000 110616 100000          -D+C->T  S34->M
0437 003000 006700 130761 D+C->Q   T3->D(A5) 0761 X M   S3->M
0440 073000 003611 103702          D+C->S   M->D   3702 X M, S73->M
0441 145000 003700 003632          S->S45   D+C->Q   M->D, 3632 X M
0442 000000 113600 000000 (***************************)-D+C->Q, M->D
0443 033000 110616 100000 S33->M                        (*)-D+C->T
0444 014000 006700 133647 3647 X M S14->M        (*)D+C->Q, T3->D(A27)
0445 046000 003611 101122 M->D      1122 X M S46->M  (*)D+C->S
0446 173000 003700 003632 D+C->Q   M->D      3632 X M(*)S->S73
                                                      (***************
0447 000000 003600 000000          D+C->Q   M->D
0450 033000 110616 100000          -D+C->T  S33->M
0451 014000 006700 131122 D+C->Q   T3->D(A6) 1122 X M   S14->M
0452 072000 003611 103647          D+C->S   M->D   3647 X M, S72->M
0453 146000 003700 003632          S->S46   D+C->Q   M->D, 3632 X M
0454 000000 113600 000000 (***************************)-D+C->Q, M->D
0455 024000 110616 100000 S24->M                        (*)-D+C->T
0456 013000 006700 133610 3610 X M S13->M        (*)D+C->Q, T3->D(A26)
0457 047000 003611 101261 M->D      1261 X M S47->M  (*)D+C->S
0460 172000 003700 003632 D+C->Q   M->D      3632 X M(*)S->S72
                                                      (***************
0461 000000 003600 000000          D+C->Q   M->D
```

```
0462 024000 110616 100000             -D+Q->T   S24->M
0463 013000 006700 131261 D+Q->Q      T3->D(A7) 1261 X M    S13->M
0464 071000 003611 103610             D+Q->S    M->D    3610 X M, S71->M
0465 147000 003700 003632             S->S47    D+Q->Q     M->D, 3632 X M
0466 000000 113600 000000 (***************************)-D+Q->Q, M->D
0467 000000 110616 100000 S0 ->M                      (*)-D+Q->T
0470 025000 006700 133544 3544 X M S25->M        (*)D+Q->Q, T3->D(A25)
0471 050000 003611 101417 M->D        1417 X M S50->M (*)D+Q->S
0472 171000 003700 003632 D+Q->Q      M->D        3632 X M(*)S->S71
                                                      (***************
0473 000000 003600 000000             D+Q->Q    M->D
0474 000000 110616 100000             -D+Q->T   S0->M
0475 025000 006700 131417 D+Q->Q      T3->D(A8) 1417 X M    S25->M
0476 070000 003611 103544             D+Q->S    M->D    3544 X M, S70->M
0477 150000 003700 003632             S->S50    D+Q->Q     M->D, 3632 X M
0500 000000 113600 000000 (***************************)-D+Q->Q, M->D
0501 026000 110616 100000 S26->M                       (*)-D+Q->T
0502 011000 006700 133473 3473 X M S11->M        (*)D+Q->Q, T3->D(A24)
0503 051000 003611 101553 M->D        1553 X M S51->M  (*)D+Q->S
0504 170000 003700 003632 D+Q->Q      M->D        3632 X M(*)S->S70
                                                      (***************
0505 000000 003600 000000             D+Q->Q    M->D
0506 026000 110616 100000             -D+Q->T   S26->M
0507 011000 006700 131553 D+Q->Q      T3->D(A9) 1553 X M    S11->M
0510 067000 003611 103473             D+Q->S    M->D    3473 X M, S67->M
0511 151000 003700 003632             S->S51    D+Q->Q     M->D, 3632 X M
0512 000000 113600 000000 (***************************)-D+Q->Q, M->D
0513 031000 110616 100000 S31->M                       (*)-D+Q->T
0514 016000 006700 133416 3416 X M S16->M        (*)D+Q->Q, T3->D(A23)
0515 052000 003611 101705 M->D        1705 X M S52->M  (*)D+Q->S
0516 167000 003700 003632 D+Q->Q      M->D        3632 X M(*)S->S67
                                                      (***************
0517 000000 003600 000000             D+Q->Q    M->D
0520 031000 110616 100000             -D+Q->T   S31->M
0521 016000 006700 131705 D+Q->Q      T3->D(A10) 1705 X M   S16->M
0522 066000 003611 103416             D+Q->S    M->D    3416 X M, S66->M
0523 152000 003700 003632             S->S52    D+Q->Q     M->D, 3632 X M
0524 000000 113600 000000 (***************************)-D+Q->Q, M->D
0525 036000 110616 100000 S36->M                       (*)-D+Q->T
0526 001000 006700 133334 3334 X M S1 ->M        (*)D+Q->Q, T3->D(A22)
0527 053000 003611 102034 M->D        2034 X M S53->M  (*)D+Q->S
0530 166000 003700 003632 D+Q->Q      M->D        3632 X M(*)S->S66
                                                      (***************
0531 000000 003600 000000             D+Q->Q    M->D
0532 036000 110616 100000             -D+Q->T   S36->M
0533 001000 006700 132034 D+Q->Q      T3->D(A11) 2034 X M   S1->M
0534 065000 003611 103334             D+Q->S    M->D    3334 X M, S65->M
0535 153000 003700 003632             S->S53    D+Q->Q     M->D, 3632 X M
0536 000000 113600 000000 (***************************)-D+Q->Q, M->D
0537 002000 110616 100000 S2->M                        (*)-D+Q->T
0540 006000 006700 133246 3246 X M S6->M         (*)D+Q->Q, T3->D(A21)
0541 054000 003611 102161 M->D        2161 X M S54->M  (*)D+Q->S
0542 165000 003700 003632 D+Q->Q      M->D        3632 X M(*)S->S65
                                                      (***************
0543 000000 003600 000000            D+C->C   M->D
0544 002000 110616 100000            -D+C->T  S2->M
0545 006000 006700 132161 D+C->C     T3->D(A12) 2161 X M   S6->M
0546 064000 003611 103246            D+C->S   M->D   3246 X M, S64->M
0547 154000 003700 003652            S->S54   D+C->C    M->D, 3632 X M
0550 000000 113600 000000 (***************************)-D+C->C, M->D
0551 032000 110616 100000 S32->M                   (*)-D+C->T
0552 005000 006700 133154 3154 X M S5 ->M       (*)D+C->C, T3->D(A20)
0553 055000 003611 102303 M->D       2303 X M S55->M  (*)D+C->S
0554 164000 003700 003632 D+C->C     M->D      3632 X M(*)S->S64
                                                   (***************

0555 000000 003600 000000            D+C->C   M->D
0556 032000 110616 100000            -D+C->T  S32->M
0557 005000 006700 132303 D+C->C     T3->D(A13) 2303 X M   S5->M
0560 063000 003611 103154            D+C->S   M->D   3154 X M, S63->M
0561 155000 003700 003632            S->S55   D+C->C    M->D, 3632 X M
0562 000000 113600 000000 (***************************)-D+C->C, M->D
0563 035000 110616 100000 S35->M                   (*)-D+C->T
0564 012000 006700 133057 3057 X M S12->M       (*)D+C->C, T3->D(A19)
0565 056000 003611 102423 M->D       2423 X M S56->M  (*)D+C->S
0566 163000 003700 003632 D+C->C     M->D      3632 X M(*)S->S63
                                                   (***************

0567 000000 003600 000000            D+C->C   M->D
0570 035000 110616 100000            -D+C->T  S35->M
0571 012000 006700 132423 D+C->C     T3->D(A14) 2423 X M   S12->M
0572 062000 003611 103057            D+C->S   M->D   3057 X M, S62->M
0573 156000 003700 003632            S->S56   D+C->C    M->D, 3632 X M
0574 000000 113600 000000 (***************************)-D+C->C, M->D
0575 022000 110616 100000 S22->M                   (*)-D+C->T
0576 015000 006700 132755 2755 X M S15->M       (*)D+C->C, T3->D(A18)
0577 057000 003611 102537 M->D       2537 X M S57->M  (*)D+C->S
0600 162000 003700 003632 D+C->C     M->D      3632 X M(*)S->S62
                                                   (***************

0601 000000 003600 000000            D+C->C   M->D
0602 022000 110616 100000            -D+C->T  S22->M
0603 015000 006700 132537 D+C->C     T3->D(A15) 2537 X M   S15->M
0604 061000 003611 102755            D+C->S   M->D   2755 X M, S61->M
0605 157000 003700 003632            S->S57   D+C->C    M->D, 3632 X M
0606 000000 113600 000000                                -D+C->C, M->D
0607 000000 110616 000000                                -D+C->T
0610 000000 006707 030000 JUMP                     D+C->C, T3->D(A17)
0611 000000 000700 000000 D+C->C (NOP)
                           (AT BEGINNING OF PROGRAM:    D+C->S)
                           (AT BEGINNING OF PROGRAM:    S->S61)

     000000 170010 000000 END
```

```
                    (ASSEMBLED 9-28-77)

                    (MICROCODE FOR INVERSE DPCM • DCT)
                    (TAPE 1, 9-28-77)

                    (INVERSE DPCM)

                    (S60->M AT END)
0000 000000 000200 000000 0+Q->Q (NO OPERATION)

0001 000000 005010 003632 3632 X M  RIN->D    (*)
                          (********)          (*********)
0002 040000 003700 100000 S40->M  (*) M->D       D+0->Q(*)
                                 (***********)       (*)
0003 000000 005611 003632 3632 X M RIN->D     (*)D+Q->S(*)
                          (********)          (*********)
0004 041000 003700 100000 S41->M(*)M->D          D+0->Q (*)
                                (**********)          (*******)
0005 160000 005631 003632 3632 X M  RIN->D (*)D+Q->S D+0->RQ(*)S->S60
                          (**********)      (************)
0006 042000 003700 100000 S42->M      (*)M->D     D+0->Q (*)
                                   (**********)         (*)
0007 140000 005611 003632 3632X M RIN->D        (*)D+Q->S (*) S->S40
                          (*******)             (********)
0010 043000 003700 100000 S43->M (*)M->D       D+0->Q  (*)
                                (*********)            (*)
0011 141000 005611 003632 3632X M RIN->D (*)  D+Q->S  (*)S->S41
                          (******)         (**********)  (********)
0012 044000 003700 100000 S44->M(*) M->D        D+0->Q (*)
                                 (********)           (*)
0013 142000 005611 003632 3632X M RIN->D  (*)D+Q->S (*) S->S42
                          (******)          (********) (********)
0014 045000 003700 100000 S45->M (*) M->D      D+0->Q(*)
                                (********)           (*)
0015 143000 005611 003632 3632 X M  RIN->D (*)D+Q->S (*) S->S43
                          (********)        (********) (********)
0016 073000 003700 100000 S73->M (*)  M->D     D+0->Q(*)
                                (*********)          (*)
0017 144000 005611 003632 3632 X M RIN->D(*) D+Q->S (*) S->S44
                          (********)       (*********) (********)
0020 046000 003700 100000 S46->M (*) M->D     D+0->Q(*)
                                (********)          (**)
0021 145000 005611 003632 3632 X M RIN->D(*) D+Q->S (*) S->S45
                          (*******)        (*********)  (********)
0022 072000 003700 100000 S72 ->M(*) M->D     D+0->Q(*)
                                (********)          (*)
0023 173000 005611 003632 3632 X M RIN->D(*) D+Q->S (*) S->S73
                          (********)      (********) (********)
0024 047000 003700 100000 S47->M (*) M->D      D+0->Q(*)
                                (********)          (**)
0025 146000 005611 003632 3632 X M  RIN->D(*) D+Q->S (*)S->S46
                          (********)       (********) (*******)
```

```
0026 071000 003700 100000 S71->M (*) M->D     D+C->Q(*)
                                (********)        (**)
0027 172000 005611 003632 3632 X M RIN->D(*)  D+Q->S (*)S->S72
                          (********)       (********)  (********)
0030 050000 003700 100000 S50->M (*) M->D     D+C->Q(*)
                                (********)        (**)
0031 147000 005611 003632 3632X M RIN->D (*)  D+Q->S (*) S->S47
                          (*******)       (*********) (********)
0032 070000 003700 100000 S70->M(*) M->D      D+C->Q(*)
                               (********)        (**)
0033 171000 005611 003632 3632X M RIN->D (*)  D+Q->S (*) S->S71
                          (*******)       (*********) (********)
0034 051000 003700 100000 S51->M(*) M->D    D+C->Q(*)
                                (********)      (**)
0035 150000 005611 003632 3632X M RIN->D(*) D+Q->S (*) S->S50
                          (*******)       (********) (********)
0036 067000 003700 100000 S67->M  (*) M->D  D+C->Q(*)
                                (********)      (**)
0037 170000 005611 003632 3632 X M  RIN->D (*)D+Q->S (*)S->S70
                          (********)        (********) (********)
0040 052000 003700 100000 S52->M (*) M->D    D+C->Q(*)
                                  (*********)      (**)
0041 151000 005611 003632 3632 X M  RIN->D (*)D+Q->S (*)S->S51
                          (********)        (*********) (********)
0042 066000 003700 100000 S66->M (*)  M->D     D+C->Q(*)
                                (*********)       (**)
0043 167000 005611 003632 3632X M  RIN->D(*)  D+Q->S (*)S->S67
                          (********)       (********)  (********)
0044 053000 003700 100000 S53->M (*) M->D    D+C->Q(*)
                                   (*********)      (**)
0045 152000 005611 003632 3632 X M  RIN->D  (*)D+Q->S (*) S->S52
                          (********)        (********)(********)
0046 065000 003700 100000 S65->M(*)  M->D      D+C->Q(*)
                                (**********)      (**)
0047 166000 005611 003632 3632X M RIN->D (*)   D+Q->S  (*)S->S66
                                                        (********)

                          (********)          (********)
0050 054000 003700 100000 S54->M (*)  M->D      D+C->Q(*)
                                (*********)       (*)
0051 153000 005611 003632 3632 X M  RIN->D (*)D+C->S (*)S->S53
                          (********)        (********) (*******)
0052 064000 003700 100000 S64->M (*)  M->D    D+C->Q (*)
                                (*********)       (**)
0053 165000 005611 003632 3632 X M RIN->D (*) D+Q->S  (*)S->S65
                          (********)        (********) (*******)
0054 055000 003700 100000 S55->M (*)M->D      D+C->Q(*)
                                (*********)

0055 154000 005611 003632 3632 X M RIN->D (*) D+Q->S (*) S->S54
                          (*******)       (********)  (********)
0056 063000 003700 100000 S63->M (*) M->D     D+C->Q(*)
                                (*********)

0057 164000 005611 003632 3632 X M RIN->D (*) D+Q->S (*) S->S64
```

```
                          (*******)        (*******)   (******)
0060 056000 003700 100000 S56->M (*) M->D      D+0->Q (*)
                                 (*******)           (***)
0061 155000 005611 003632 3632 X M  RIN->D (*) D+Q->S (*)S->S55
                          (*******)       (*********) (*******)
0062 062000 003700 100000 S62->M (*)  M->D     D+0->Q (*)
                                   (*******)         (**)
0063 163000 005611 003632 3632 X M  RIN->D (*) D+Q->S (*)S->S63
                          (********)          (*******) (******)
0064 057000 003700 100000 S57->M (*)  M->D     D+0->Q (*)
                                 (*******)            (**)
0065 156000 005611 003632 3632 X M  RIN->D (*) D+Q->S (*)S->S56
                          (********)        (*********) (******)
0066 061000 003700 100000 S61->M (*)  M->D      D+0->Q (*)
                                 (**********)         (**)
0067 162000 005611 003632 3632 X M  RIN->D (*)D+Q->S (*)S->S62
                          (********)         (*******) (********)
0070 060000 003700 100000 S60->M (*)  M->D    D+0->Q (*)
                                  (*********)         (*)
0071 157000 000611 001324 1324 X M          (*) D+Q->S (*)S->S57
                                            (********) (*******)
                                                  (S->S61 BELOW)

                          (INVERSE DPCM AND DCT)
                          (TAPE 2, 9-28-77)


                          (INVERSE FINAL ROTATION)
                                                    (S60->M ABOVE)
                                                    (2650X M ABOVE)
0072 041000 003010 100000 S41->M                    (*) M->D
0073 161000 000700 000777 0777 X M          (*) S->S61   (*) D+0->Q
                                            (*************)
0074 042000 003711 100031 M->D       0031X M     S42->M  (*) D+0->S
0075 100001 003730 000776 D+0->R1     M->D    0776X M   (*) S->S0
                          (********)                (****************)
0076 043000 003711 100062 S43->M (*) D+0->S    M->D      0062X M
0077 101002 003730 000772 0772X M(*) S->S1      D+0->R2    M->D
                                (***********************)
0100 044000 003711 100113 M->D       0113X M     S44->M  (*) D+0->S
0101 102005 003730 000766 D+0->R3     M->D    0766X M   (*) S->S2
                          (********)                (****************)
0102 073000 003711 100144 S73->M (*) D+0->S    M->D       0144X M
                                (**********)
0103 045004 003730 100174 0174X M    S45->M(*) D+0->R4      M->D
0104 103000 003711 000761 M->D       0761X M(*) S->S3       D+0->S
                                            (****************************)

0105 073000 003700 100174 D+0->Q     M->D        0174X M     S73->M
                          (********)
0106 072005 003630 100761 S72->M (*)D+Q->R5    M->D        0761 X M
                                (***********)
0107 046000 003700 100225 0225X M    S46->M (*)D+0->Q     M->D
```

```
                                                   (*)          (*********)
0110 104000 113611 000752   M->D     0752X M(*)-D+C->S(*)S->S4
                                                   (********************)
0111 072000 003700 100226 D+C->D     M->D      0226X M    S72->M
                          (*********)
0112 071000 003600 100752 S71->M(*)D+C->R6    M->D          0752 X M
                                     (***********)
0113 047000 003700 100254 0254X M   S47->M (*)D+C->C    M->D
                                                   (*)          (********)
0114 105000 113611 000742   M->D     0742X M(*)-D+C->S(*)S->S5
                                                   (*******************)
0115 071000 003700 100254 D+C->C     M->D      0254X M    S71->M
                          (*********)
0116 070007 003630 100742 S70->M (*)D+C->R7   M->D          0742 X M
                                     (***********)
0117 050000 003700 100504 0504X M    S50->M (*)D+C->C    M->D
                                                   (*)          (*********)
0120 106000 113611 000731   M->D     0731X M(*)-D+C->S(*)S->S6
                                                   (*******************)
0121 070000 003700 100504 D+C->C     M->D      0504X M    S70->M
                          (*********)
0122 067010 003650 100731 S67->M (*)D+C->R10  M->D          0731 X M
                                     (***********)
0123 051000 003700 100533 0533X M    S51->M (*)D+C->C    M->D
                                                   (*)          (*********)
0124 107000 113611 000717   M->D     0717X M(*)-D+C->S  (*)S->S7
                                                   (*******************)
0125 067000 003700 100733 D+C->C     M->D      0533X M    S67->M
                          (*********)
0126 066011 003630 100717 S66->M (*)D+C->R11   M->D          0717 X M
                                     (***********)
0127 052000 003700 100561 0561X M    S52->M (*)D+C->C    M->D
                                                   (*)          (*********)
0130 110000 113611 000704   M->D     0704X M(*)-D+C->S(*)S->S10
                                                   (*******************)
0131 066000 003700 100561 D+C->C     M->D      0561X M    S66->M
                          (*********)
0132 065012 003630 100704 S65->M(*) D+C->R12  M->D          0704 X M
                                     (***********)
0133 053000 003700 100407 0407X M    S53->M(*)D+C->C     M->D
                                                   (*)          (********)
0134 111000 113611 000667   M->D     0667X M(*)-D+C->S(*)S->S11
                                                   (*******************)
0135 065000 003700 100407 D+C->C     M->D      0407X M    S65->M
                          (*********)
0136 064013 003630 100667 S64->M (*)D+C->R13  M->D          0667 X M
                                     (***********)
0137 054000 003700 100464 0464X M    S54->M (*) D+C->C     M->D
                                                   (*)          (*********)
0140 112000 113611 000652   M->D     0652X M(*)-D+C->S(*)S->S12
                                                   (*******************)
0141 064000 003700 100464 D+C->C     M->D      0464X M    S64->M
                          (*********)
0142 063014 003630 100652 S63->M (*)D+C->R14  M->D          0652 X M
```

```
                                   (*********)
0143 055000 005700 100461 0461Y M    S55->M (*)D+C->C       ->D
                                            (*)          (*********)
0144 115000 113611 000663 M->D        0663Y M(*)-D+C->S(*)S->S13
                                            (********************)
0145 063000 123700 100461 D-C->C     M->D       0461Y M    S6.->M
                        (********)
0146 062010 000600 100663 S62->M (*)D+C->R15  M->D         0663 Y M
                                   (*********)
0147 056000 003700 100505 0505Y M S56->M (*)D+C->C     M->D
                                            (*)          (*******)
0150 114000 113611 000614  M->D        0614Y M(*)-D+C->S(*)S->S14
                                            (********************)
0151 067000 003700 100505 D+C->C     M->D       0505Y M    S62->M
                        (********)
0152 061010 000600 100614 S61->M (*)D+C->R16  M->D         0614 Y M
                                   (**********)
0153 057000 003700 100560 0560Y M S57->M (*) D+C->C     M->D
                                            (*)          (********)
0154 115000 113611 000573 M->D        0573Y M(*)-D+C->S(*)S->S16
                                            (*****************)
0155 061000 004700 100560 D+C->C        ->D       0560Y M    S61->M
                        (********)
0156 040017 000600 100573 S+C->M    (*)D+C->R17      ->D          0573 Y M
                              (***********)
0157 000000 000700 001000 1000 Y M          (*)D+C->C     M->D
                                            (*)          (********)
0160 116000 113611 000000  ->D              (*)-D+C->S(*)S->S16
                                                         (*******)
0161 117000 000750 000000 D+C->RC                 S->S17


                        (INVERSE DPCM AND DCT)
                        (TAPE 3, 9-15-77)


                        (FIRST COLUMN OF INVERSE BUTTERFLIES)

0162 000210 001111 000000 S0->D      R10+R10->S
0163 100010 110530 000000 -D+S0->R10 S->S0
0164 010000 001530 000000 D+R0->R0   S10->D
0165 000000 110700 000000 -D+C->C
                        (********)
0166 017000 111611 000000 S17->M  (*)-D+C->S
                                 (********)
0167 001000 001700 000000 D+C->C     S1->D
0170 110000 000613 000000                D+C->M(*)(*)S->S10
                                         (***********)
0171 000000 120611 000620            0620 Y M      D-C->S
0172 101037 123113 000731 0731 Y M M->D          S->S1        S1-S17->M
0173 000000 003711 000620 M->D        D+C->M     0620 Y M
0174 137000 003700 000731 D+C->C      S->S37     M->D         0731 Y M
```

```
0175 000000 003611 000000 D+Q->S                          M->D
0176 037000 001700 000000           S37->D             D+0->Q
0177 117361 000130 000000 S->S17    R17+R1->R1
                          (*****************)
0200 016017 111630 000000 S16-> D            (*)          -D+Q->R17
                                        (*********************)
0201 002000 001700 000000 D+0->Q    S2->D
0202 000000 000613 000000          D+Q->M
0203 000000 120611 001420          1420 X M D-Q->S
0204 102056 123113 003544 3544 X M  M->D     S->S2    R2-R16->M
0205 000000 003711 001420 M->D      D+0->S    1420 X M
0206 137000 003700 003544 D+0->Q    S->S37   M->D      3544 X M
0207 000000 003611 000000 D+Q->S                        M->D
0210 037000 001700 000000           S37->D              D+0->Q
0211 116342 000130 000000 S->S16    R16+R2->R2
                          (******************)
0212 015016 111630 000000 S15->D             (*)          -D+Q->R16
                                        (*********************)
0213 003000 001700 000000 D+0->Q    S3->D
0214 000000 000613 000000           D+Q->M
0215 000000 120611 002162           2162 X M  D-Q->S
0216 103075 123113 003247 3247 X M  M->D     S->S3     R3-R15->M
0217 000000 003711 002162 M->D      D+0->S    2162 X M
0220 137000 003700 003247 D+0->Q    S->S37   M->D      3247 X M
0221 000000 003611 000000 D+Q->S                        M->D
0222 037000 001700 000000           S37->D              D+0->Q
0223 115323 000130 000000 S->S15    R15+R3->R3
                          (*****************)
0224 014015 111630 000000 S14->D             (*)          -D+Q->R15
                                        (**********************)
0225 004000 001700 000000 D+0->Q    S4->D
0226 000000 000613 000000          D+Q->M
0227 000000 120611 002650          2650 X M  D-Q->S
0230 104114 123113 002650 2650 X M  M->D     S->S4    R4-R14->M
0231 000000 003711 002650 M->D      D+0->S    2650 X M
0232 137000 003700 002650 D+0->Q    S->S37   M->D      2650 X M
0233 000000 003611 000000 D+Q->S                        M->D
0234 037000 001700 000000           S37->D          D+0->Q
0235 114304 000130 000000 S->S14    R14+R4->R4
                          (***************)
0236 013014 111630 000000 S13->D            (*)       -D+Q->R14
0237 005000 001700 000000 D+0->Q    S5->D (*)
                                       (*********************)
0240 000000 000613 000000          D+Q->M
0241 000000 120611 003247           3247 X M  D-Q->S
0242 105133 123113 002162 2162 X M  M->D     S->S5     R5-R13->M
0243 000000 003711 003247 M->D      D+0->S    3247 X M
0244 137000 003700 002162 D+0->Q    S->S37   M->D      2162 X M
0245 000000 003611 000000 D+Q->S                        M->D
0246 037000 001700 000000          S37->D              D+0->Q
0247 113265 000130 000000 S->S13    R13+R5->R5
                          (***************)
0250 012013 111630 000000 S12->D            (*)       -D+Q->R13
                                     (*********************)
```

```
0251 006000 001700 000000 D+Q->Q    S6->D
0252 000000 000613 000000          D+Q->M
0253 000000 120611 003544          3544 X M  D-Q->S
0254 106152 123113 001420 1420 X M  M->D      S->S6     R6-R12->M
0255 000000 003711 003544 M->D      D+Q->S    3544 X M
0256 137000 003700 001420 D+Q->Q    S->S37    M->D      1420 X M
0257 000000 003611 000000 D+Q->S                        M->D
0260 037000 001700 000000           S37->D        D+Q->Q
0261 112246 000130 000000 S->S12    R12+R6->R6
                          (****************)
0262 011012 111630 000000 S11->D         (*)          -D+Q->R12
                                         (**********************)
0263 007000 001700 000000 D+Q->Q    S7->D
0264 000000 000613 000000           D+Q->M
0265 000000 120611 003731          3731 X M  D-Q->S
0266 107171 123113 000620 0620 X M  M->D      S->S7     R7-R11->M
0267 000000 003711 003731 M->D      D+Q->S    3731 X M
0270 137000 003700 000620 D+Q->Q    S->S37    M->D      0620 X M
0271 000000 003611 000000 D+Q->S                        M->D
0272 037000 001700 000000          S37->D               D+Q->Q

0273 111227 000130 000000 S->S11    R11+R7->R7
                          (*****************)
0274 000011 110630 000000                (*)          -D+Q->R11
                                         (**********************)

                          (SECOND COLUMN OF INVERSE BUTTERFLIES)
0275 000104 001111 000000 S0->D     R4+R4->S
0276 100004 110530 000000 -D+R0->R4  S->S0
0277 004000 001530 000000 D+R0->R0  S4->D
0300 000000 110700 000000 -D+Q->Q
                          (********)
0301 007000 111611 000000 S7->D  (*)-D+Q->S
                                 (*********)
0302 001000 001700 000000 D+Q->Q    S1->D(*)
0303 104000 000613 000000          D+Q->M(*)   S->S4
                                        (**********************)
0304 000000 120611 001420          1420 X M   D-Q->S
0305 101027 123113 003544 3544 X M   M->D      S->S1     R1-R7->M
0306 000000 003711 001420 M->D      D+Q->S    1420 X M
0307 137000 003700 003544 D+Q->Q    S->S37    M->D      3544 X M
0310 000000 003611 000000 D+Q->S                        M->D
0311 037000 001700 000000           S37->D              D+Q->Q
0312 107101 000130 000000 S->S7     R7+R1->R1
                          (***************)
0313 006007 111630 000000 S6->D          (*)         -D+Q->R7
                                        (***********************)
0314 002000 001700 000000 D+Q->Q    S2->D
0315 000000 000613 000000          D+Q->M
0316 000000 120611 002650          2650 X M  D-Q->S
0317 102046 123113 002650 2650 X M  M->D      S->S2     R2-R6->M
0320 000000 003711 002650 M->D      D+Q->S    2650 X M
0321 137000 003700 002650 D+Q->Q    S->S37    M->D      2650 X M
```

```
0322 000000 000011 000000 D+C->R                                 M->D
0323 037000 001700 000000            S37->D                       D+C->C
0324 100142 000100 000000 S->S2      R2+R2->R2
                          (*********************)
0325 000000 111100 000000 S5->D                  (*)              -D+C->R
                                                 (*****************************)
0326 005000 001700 000000 D+C->C     S5->D
0327 000000 000013 000000            D+C->M
0330 000000 120011 003544            3544 X M  D-C->S
0331 105005 125113 001420 1420 Y M   M->D      S->S5           R5-R5->M
0332 000000 003711 003544 M->D       D+C->S    3544 X M
0333 137000 003700 001420 D+C->C     S->S37    M->D            1420 Y M
0334 000000 000011 000000 D+C->R                                 M->D
0335 037000 001700 000000            S37->D                       D+C->C
0336 100124 000100 000000 S->S5      R5+R5->R5
                          (*********************)
0337 000000 110100 000000                        (*)              -D+C->D
                                                 (*****************************)
0340 010014 001111 000000 S10->D     R14+R14->S
0341 000214 000000 000000 D+R10->R14
0342 014210 111000 000000 -D+R10->R10 S14->D
0343 114000 110700 000000 -D+C->C              S->S14
                          (*********)
0344 011000 111011 000000 S11->D (*)-D+C->R
                                   (*********)
0345 017000 001700 000000 D+C->C     S17->D(*)
0346 110000 000013 000000            D+C->M(*)  S->S10
                                                 (*****************************)
0347 000000 120011 001420            1420 Y M  D-C->S
0350 117011 125113 003544 3544 X M   M->D      S->S17        R17-R11->M
0351 000000 003711 001420 M->D       D+C->S    1420 Y M
0352 137000 003700 003544 D+C->C     S->S37    M->D            3544 Y M
0353 000000 000011 000000 D+C->S                                 M->D
0354 037000 001700 000000          S37->D                         D+C->C
0355 111237 000100 000000 S->S11     R11+R17->R17
                          (*********************)
0356 012011 111000 000000 S12->D                 (*)              -D+C->R11
                                                 (*****************************)
0357 016000 001700 000000 D+C->C     S16->D
0360 000000 000013 000000            D+C->M
0361 000000 120011 002000            2000 X M  D-C->S
0362 116002 125113 002050 2050 X M   M->D      S->S16          R16-R12->M
0363 000000 003711 002050 M->D       D+C->S    2050 Y M
0364 137000 003700 002000 D+C->C     S->S37    M->D            2000 Y M
0365 000000 000011 000000 D+C->R                                 M->D
0366 037000 001700 000000            S37->D                       D+C->C
0367 112236 000100 000000 S->S12     R12+R16->R16
                          (*********************)
0370 015012 111000 000000 S15->D                 (*)              -D+C->R12
                                                 (*****************************)
0371 015000 001700 000000 D+C->C     S15->D
0372 000000 000013 000000            D+C->M
0373 000000 120011 003544            3544 Y M  -C->S
0374 115002 125113 001420 1420 Y     M->D      S->S15          R15-R15->M
```

```
0376 000000 003711 003544 M->D        M+C->S      2650 X M
0376 157000 003700 001400 D+C->C      S->S37      M->D        2650 X M
0377 000000 003611 000000 D+C->S                              S->D
0400 037000 001700 000000             S37->D                  D+C->C
0401 115276 000130 000000 S->S15      M15+M15->M15
                          (*******************)
0402 000016 110660 000000                         (*)         -D+C->M15
                                                  (*********************)

                          (INVERSE PROG AND ROLL)
                          (TAPE 6, 5-15-77)

                          (40 DECAMA BUTTERFLIES)

0403 000042 001111 000000 S0->D       M1+M2->M2
0404 100002 110550 000000 -D+M0->M0   S->M0
0405 002000 001550 000000 D+M0->M0    S2->D
0406 000000 110700 000000 -D+C->C
                          (*******)
0407 003000 111611 000000 S3->D (*) -D+C->S
                                (***********)
0410 001000 001700 000000 D+C->C      S1->D   (*)
0411 102000 000616 000000             D+C->M  (*)    C->S2
                                              (*************************)
0412 000000 120611 002650             2650 X M        D-C->S
0413 101025 125113 002650 2650 X M    M->D            S->S1       M1-M5->M5
0414 000000 003711 002650 M->D        D+C->S          2650 X M
0415 157000 003700 002650 D+C->C      S->S37          M->D        2650 X M
0416 000000 003611 000000 D+C->S                                  S->D
0417 037000 001700 000000             S37->D                      D+C->C
0420 103001 000130 000000 S->S3       M3+M1->M1
                          (*****************)
0421 000003 110630 000000                     (*)                 -D+C->M3
                                              (********************)
0422 004000 003111 010000 S4->D       M0+M7->M7
0423 007104 000660 000000 D+M4->M4
0424 000104 110660 000000 -D+M4->M4   C->S
0425 100000 110700 000000 -D+C->C                  S->S6
                          (*********)
0426 006000 111611 000000 S6->D   (*)-D+C->S
                                  (*********)
0427 007000 001700 000000 D+C->C      S7->D(*)
0430 104000 000616 000000             D+C->M(*)  S->S4
                                                 (*************************)
0431 000000 120611 002650             2650 X M   D-C->S
0432 107165 125113 002650 2650 X M    M->D       S->S7       M7-M5->M5
0433 000000 003711 002650 M->D        D+C->S     2650 X M
0434 157000 003700 002650 D+C->C      S->S37     M->D        2650 X M
0435 000000 003611 000000 D+C->S                             S->D
0436 037000 001700 000000             S37->D                 D+C->C
```

```
0437 105127 000130 000000 S->S5     R5+R7->R7
                          (****************)
0440 000005 110630 000000                  (*)          -D+0->R5
                                           (**********************)
0441 014356 001111 000000 S14->D     R16+R16->S
0442 000316 000530 000000 D+R14->R16
0443 016314 111530 000000 -D+R14->R14  S16->D
0444 116000 110700 000000 -D+0->Q                  S->S16
                          (********)
0445 015000 111611 000000 S15->D (*)  -D+Q->S
                                 (********)
0446 017000 001700 000000 D+0->Q     S17->D(*)
0447 114000 000613 000000            D+0->M(*)      S->S14
                                            (**************************)
0450 000000 120611 002650            2650 X M  D-Q->S
0451 117375 123113 002650 2650 X M  M->D        S->S17     R17-R15->M
0452 000000 003711 002650 M->D       D+0->S     2650 X M
0453 137000 003700 002650 D+0->Q     S->S37     M->D         2650 X M
0454 000000 003611 000000 D+Q->S                             M->D
0455 037000 001700 000000            S37->D                  D+0->Q
0456 115337 000130 000000 S->S15     R15+R17->R17
                          (****************)
0457 000015 110630 000000                    (*)          -D+0->R15
                                             (*********************)
0460 010252 001111 000000 S10->D     R12+R12->S
0461 110212 110530 000000 -D+R10->R12 S->S10
0462 012210 001530 000000 D+R10->R10 S12->D
0463 000000 110700 000000 -D+0->Q
                          (********)
0464 013000 111611 000000 S13->D  (*)-D+Q->S
                                  (*********)
0465 011000 001700 000000 D+0->Q     S11->D(*)
0466 112000 000613 000000            D+Q->M(*) S->S12
                                              (*********************)
0467 000000 120611 002650           2650 X M  D-Q->S
0470 111233 123113 002650 2650 X M  M->D        S->S11     R11-R13->M
0471 000000 003711 002650 M->D       D+0->S     2650 X M
0472 137000 003700 002650 D+0->Q     S->S37     M->D         2650 X M
0473 000000 003611 000000 D+Q->S                             M->D
0474 037000 001700 000000            S37->D                  D+0->Q
0475 113271 000130 000000 S->S13     R13+R11 ->R11
                          (*****************)
0476 001013 111630 000000 S1->D                (*)          -D+0->R13
                                               (***********************)

                          (SCALING)

                          (S1->D  ABOVE)
0477 000000 000700 000000 D+0->Q
0500 003000 001611 000000 D+Q->S (*) S3->D
0501 101000 000700 000000 S->S1 (*)  D+0->Q
                          (*******)
0502 005000 001611 000000 S5->D (*) D+Q->S
0503 103000 000700 000000 D+0->Q (*)S->S3
```

```
                                        (********)
0504 007000 001611 000000 D+Q->S(*) S7->D
0505 105000 000700 000000 S->S5 (*) D+Q->Q
                                    (*******)
0506 011000 001611 000000 S11->D (*) D+Q->S
0507 107000 000700 000000 D+Q->Q(*) S->S7
                                        (*********)
0510 013000 001611 000000 D+Q->S(*) S13->D
0511 111000 000700 000000 S->S11(*) D+Q->Q
                                    (*******)
0512 015000 001611 000000 S15->D(*) D+Q->S
0513 113000 000700 000000 D+Q->Q(*) S->S13
                                        (*********)
0514 017000 001611 000000 D+Q->S(*) S17->D
0515 115000 000700 000000 S->S15(*) D+Q->Q
                                    (*******)
0516 000000 000611 000000           (*) D+Q->S
0517 117021 000130 000000 R1+R1->R1(*) S->S17
                                          (**********)
0520 000063 000130 000000 R3+R3->R3
0521 000125 000130 000000 R5+R5->R5
0522 000167 000130 000000 R7+R7->R7
0523 000251 000130 000000 R11+R11->R11
0524 000273 000130 000000 R13+R13->R13
0525 000335 000130 000000 R15+R15->R15
0526 000377 001130 000000 R17+R17->R17 (*) S0->D

                          (SUMS, DIFFERENCES AND OUTPUT)

                          (S0->D ABOVE)
0527 000000 000500 000000 D+R0->Q
0530 012020 111011 000000 -R1+Q->S  S12--D
0531 013040 111500 000000               -D+R12->Q S13->D
0532 137000 000615 000000 S->S37      D+Q->ROUT
0533 000000 000614 000000             D+Q->LOUT
                          (*********)
0534 016000 111611 000000 S16->D     (*)-D+Q->S
0535 136340 000500 000000 D+R16->Q(**) S->S36
                                           (*********************)
0536 000360 000015 000000 R17+Q->ROUT

0537 004360 111011 000000 -R17+Q->S  S4->D
0540 005100 111500 000000                -D+R4->Q  S5->D
0541 135000 000614 000000 S->S35     D+Q->LOUT
                          (********)
0542 006000 111611 000000 S6->D  (*)-D+Q->S
0543 134140 000500 000000 D+R6 ->Q(*)S->S34
                                         (*********************)
0544 000160 000015 000000 R7+Q-> ROUT
0545 014160 111011 000000 -R7+Q->S     S14->D
0546 015300 111500 000000                -D+R14->Q  S15->D
0547 133000 000614 000000 S->S33       D+Q->LOUT
                          (*********)
```

```
0550 010000 111611 000000 S10->D   (*)-D+C->S
0551 132200 000600 000000 D+R10->C(*)S->S32
                              (**********************)
0552 000220 000015 000000 R11+C-> ROUT
0553 002220 111011 000000 -R11+C->S     S2->D
0554 003040 111600 000000           -D+R2->C   S3->D
0555 131000 000614 000000 S->S31      D+C->LOUT
                          (********)
0556 002000 111611 000000 S2->D   (*)-D+C->S
0557 130040 000600 000000 D+R2->C(*) S->S30
                              (************************)
0560 000060 000015 000000 R3+C->ROUT
0561 010060 111011 000000 -R3+C->S      S10->D
0562 011200 111600 000000            -D+R10->C    S11->D
0563 127000 000614 000000 S->S27      D+C->LOUT
                          (*********)
0564 014000 111611 000000 S14->D  (*)-D+C->S
0565 126600 000600 000000 D+R14->C(*) S->S26
                              (************************)
0566 000340 000015 000000 R15+C->ROUT
0567 006320 111011 000000 -R15+C->S     S6->D
0570 007140 111600 000000            -D+R6->C    S7->D
0571 125000 000614 000000 S->S25      D+C->LOUT
                          (**********)
0572 004000 111611 000000 S4->D     (*) -D+C->S
0573 124100 000600 000000 D+R4 ->C(*) S->S24
                              (**********************)
0574 000120 000015 000000 R5+C->ROUT
0575 016120 111011 000000 -R5+C->S      S16->D
0576 017040 111600 000000            -D+R16->C    S17->D
0577 123000 000614 000000 S->S23      D+C->LOUT
                          (**********)
0600 012000 111611 000000 S12->D     (*)-D+C->S
0601 122240 000600 000000 D+R12->C(*) S->S22
                               (*************)
0602 000260 000015 000000 R13+C->ROUT
0603 000260 111011 000000 -R13+C->S     S0->D
0604 001000 111600 000000            -D+R0->C   S1->D
0605 121000 000614 000000 S->S21       D+C->LOUT
                          (***********)
0606 021000 111615 000000 S21->D    (*) -D+C->ROUT
                               (*************)

0607 022000 001714 000000 D+C->LOUT     S22->D
0610 023000 001715 000000 D+C->ROUT     S23->D
0611 024000 001714 000000 D+C->LOUT     S24->D
0612 025000 001715 000000 D+C->ROUT     S25->D
0613 026000 001714 000000 D+C->LOUT     S26->D
0614 027000 001715 000000 D+C->ROUT     S27->D
0615 030000 001714 000000 D+C->LOUT     S30->D
0616 031000 001715 000000 D+C->ROUT     S31->D
0617 032000 001714 000000 D+C->LOUT     S32->D
0620 033000 001715 000000 D+C->ROUT     S33->D
0621 034000 001714 000000 D+C->LOUT     S34->D
```

```
[illegible] [illegible]000 001715 000000 B+C->[illegible]      [illegible]->[illegible]
[illegible] [illegible]000 001714 000000 [illegible]+C->[illegible]     [illegible]->[illegible]
[illegible] 057000 001715 000000 [illegible]->[illegible]      SCI->[illegible]
[illegible] 000000 000714 000000 [illegible]+C->[illegible]
[illegible] [illegible]0000 000017 100000 [illegible]            SCC->[illegible]
[illegible] 000000 000200 000000 [illegible]->[illegible] ([illegible])

      [illegible] 170016 [illegible] [illegible]
```

QUANTIZER IMAGE

```
010000    000070 000070 000070 000070 000070 000070 000070 000070
010020    000070 000070 000070 000070 000070 000070 000070 000070
010040    000070 000070 000070 000070 000070 000070 000070 000070
010060    000070 000070 000070 000070 000070 000070 000070 000070
010100    000010 000010 000010 000010 000010 000010 000010 000010    T0
010120    000010 000010 000010 000010 000010 000010 000010 000010
010140    000010 000010 000010 000010 000010 000010 000010 000010
010160    000010 000010 000010 000010 000010 000010 000010 000010
010200    000040 000040 000042 000042 000044 000044 000046 000046
010220    000050 000050 000052 000052 000054 000054 000056 000056
010240    000060 000060 000062 000062 000064 000064 000066 000066
010260    000070 000070 000072 000072 000074 000074 000076 000076    T1
010300    000000 000000 000002 000002 000004 000004 000006 000006
010320    000010 000010 000012 000012 000014 000014 000016 000016
010340    000020 000020 000022 000022 000024 000024 000026 000026
010360    000030 000030 000032 000032 000034 000034 000036 000036
010400    000057 000057 000057 000057 000057 000057 000057 000057
010420    000057 000057 000057 000057 000057 000057 000057 000057
010440    000057 000057 000057 000057 000057 000057 000057 000057
010460    000074 000074 000074 000074 000074 000074 000074 000074    T2
010500    000004 000004 000004 000004 000004 000004 000004 000004
010520    000021 000021 000021 000021 000021 000021 000021 000021
010540    000021 000021 000021 000021 000021 000021 000021 000021
010560    000021 000021 000021 000021 000021 000021 000021 000021
010600    000040 000041 000042 000043 000044 000045 000046 000047
010620    000050 000051 000052 000053 000054 000055 000056 000057
010640    000060 000061 000062 000063 000064 000065 000066 000067
010660    000070 000071 000072 000073 000074 000075 000076 000077    T3
010700    000000 000001 000002 000003 000004 000005 000006 000007
010720    000010 000011 000012 000013 000014 000015 000016 000017
010740    000020 000021 000022 000023 000024 000025 000026 000027
010760    000030 000031 000032 000033 000034 000035 000036 000037
011000    000050 000050 000050 000050 000050 000050 000050 000050
011020    000050 000050 000050 000050 000050 000050 000050 000064
011040    000064 000064 000064 000064 000064 000064 000064 000064
011060    000072 000072 000072 000072 000076 000076 000076 000076    T4
011100    000002 000002 000002 000002 000006 000006 000006 000006
011120    000014 000014 000014 000014 000014 000014 000014 000014
011140    000014 000030 000030 000030 000030 000030 000030 000030
011160    000030 000030 000030 000030 000030 000030 000030 000030
011200    000040 000041 000042 000043 000044 000045 000046 000047
011220    000050 000051 000052 000053 000054 000055 000056 000057
011240    000060 000061 000062 000063 000064 000065 000066 000067
011260    000070 000071 000072 000073 000074 000075 000076 000077    T5
011300    000000 000001 000002 000003 000004 000005 000006 000007
011320    000010 000011 000012 000013 000014 000015 000016 000017
011340    000020 000021 000022 000023 000024 000025 000026 000027
011360    000030 000031 000032 000033 000034 000035 000036 000037
```

```
011400    000043 000043 000043 000043 000043 000043 000043 000043
011420    000054 000054 000054 000054 000054 000054 000054 000062
011440    000062 000062 000062 000062 000066 000066 000066 000066
011460    000071 000071 000073 000073 000075 000075 000077 000077
011500    000001 000001 000003 000003 000005 000005 000007 000007    T6
011520    000012 000012 000012 000012 000016 000016 000016 000016
011540    000016 000024 000024 000024 000024 000024 000024 000024
011560    000035 000035 000035 000035 000035 000035 000035 000035
011600    000200 000201 000202 000203 000204 000205 000206 000207
011620    000210 000211 000212 000213 000214 000215 000216 000217
011640    000220 000221 000222 000223 000224 000225 000226 000227
011660    000230 000231 000232 000233 000234 000235 000236 000237
011700    000240 000241 000242 000243 000244 000245 000246 000247
011720    000250 000251 000252 000253 000254 000255 000256 000257
011740    000260 000261 000262 000263 000264 000265 000266 000267
011760    000270 000271 000272 000273 000274 000275 000276 000277
012000    000300 000301 000302 000303 000304 000305 000306 000307
012020    000310 000311 000312 000313 000314 000315 000316 000317
012040    000320 000321 000322 000323 000324 000325 000326 000327
012060    000330 000331 000332 000333 000334 000335 000336 000337
012100    000340 000341 000342 000343 000344 000345 000346 000347
012120    000350 000351 000352 000353 000354 000355 000356 000357
012140    000360 000361 000362 000363 000364 000365 000366 000367
012160    000370 000371 000372 000373 000374 000375 000376 000377    T7
012200    000000 000001 000002 000003 000004 000005 000006 000007
012220    000010 000011 000012 000013 000014 000015 000016 000017
012240    000020 000021 000022 000023 000024 000025 000026 000027
012260    000030 000031 000032 000033 000034 000035 000036 000037
012300    000040 000041 000042 000043 000044 000045 000046 000047
012320    000050 000051 000052 000053 000054 000055 000056 000057
012340    000060 000061 000062 000063 000064 000065 000066 000067
012360    000070 000071 000072 000073 000074 000075 000076 000077
012400    000100 000101 000102 000103 000104 000105 000106 000107
012420    000110 000111 000112 000113 000114 000115 000116 000117
012440    000120 000121 000122 000123 000124 000125 000126 000127
012460    000130 000131 000132 000133 000134 000135 000136 000137
012500    000140 000141 000142 000143 000144 000145 000146 000147
012520    000150 000151 000152 000153 000154 000155 000156 000157
012540    000160 000161 000162 000163 000164 000165 000166 000167
012560    000170 000171 000172 000173 000174 000175 000176 000177
```

MICROPROCESSOR INTERRUPT SERVICE ROUTINE

```
016000 062737  INT:    ADD  #2,@#CAMCSR
016002 000002
016004 164110
016006 052737          BIS  #2Ø, @#CAMCSR           INCREMENT CAMERA STRIPE
016010 000020
016012 164110
016014 042737          BIC  #177741, @#CAMCSR
016016 177741
016020 164110
016022 012737          MOV  #-8., @#COUNT
016024 177770
016026 016140
016030 005237  LOOPA:  INC  @#COUNT                 DELAY
016032 016140
016034 001775          BNE  LOOPA
016036 012737          MOV  #20044, @#M1SAR         CAMERA -> M1
016040 020044
016042 164012
016044 012737          MOV  #Ø, @#ITL               RESET INT TIME LIMIT
016046 000000
016050 [illegible]
016052 000137          JMP  @#LOOPB
016054 [illegible]
016056 000137  LOOPC:  JMP  @#LOOPE
016060 016340

016062 012737  QUIT:   MOV  #-8., @#COUNT
016064 177770
016066 016140
016070 005237  LOOPD:  INC  @#COUNT                 DELAY
016072 016140
016074 001775          BNE  LOOPD
016100 013737          MOV  @#CAMCSR, @#FSMCSR
016102 164110
016104 164100
016106 052737          BIS  #6Ø, @#FSMCSR           INCREMENT FSM STRIPE
016110 000060
016112 164100
016114 042737          BIC  #1777ØØ, @#FSMCSR
016116 177700
016120 164100
016122 012737          MOV  #Ø, M2SAR
016124 000000
016126 164002
016130 000401          BR   INTZ



016140 177771  COUNT:  -7
```

```
016142 012737  INT2:    MOV  #Ø, @# M1MC            MASTER CLEAR M1
016144 000000
016146 164000
016150 012737           MOV  #Ø, @# M2MC            MASTER CLEAR M2
016152 000000
016154 164002
016156 012737           MOV  #1, @# M1CSR           M1 PIXEL INPUT, WORD OUTPUT
016160 000001
016162 164022
016164 012737           MOV  #2, @# M2CSR           M2 WORD INPUT, PIXEL OUTPUT
016166 000002
016170 164062
016172 012737           MOV  #10026, @# M1DAR       M1 → M2
016174 010026
[illegible]
016200 012737           MOV  #Ø, @# M1GO            START M1
016202 000000
016204 164060
016206 012737           MOV  #100350, @# M2DAR      M2 → FSM
016210 100350
016212 164060
016214 012737           MOV  #Ø, @# M2SAR
016216 000000
016220 164052
016222 012737           MOV  #Ø, @# M2GO            START M2
016224 000000
016226 164000
016230 012737           MOV  #10044, @# M1SAR       CAMERA → M1
016232 010044
016234 164012
016236 052737           BIS  #BITØ, @# CAMCSR       START CAMERA
016240 000001
016242 164110
016244 000002           RTI

                        WAIT FOR M2 WAITING
016246 012737  LOOPE:   MOV  #Ø, @# COUNT
016250 000000
016252 [illegible]
016254 005237           INC  @# COUNT
016256 016140
016260 [illegible]      BEQ  LOOPEX                 ABANDON WAIT
016262 032737           BIT  #BIT11, @# M2CSR
016264 004000
016266 164062
016270 001766           BEQ  LOOPE
016272 000137  LOOPEX:  JMP  @# QUIT
016274 016004
```

WAIT FOR M1 WAITING

```
015500 012737  LOOPB:  MOV   #Ø, @#COUNT
015502 000000
015504 015140
015506 005237          INC   @#COUNT
015510 015140
015512 001004          BEQ   LOOPBX                 ABANDON WAIT
015514 032737          BIT   #BIT11, @#M1CSR
015516 [illegible]
015520 164022
015522 001766          BEQ   LOOPB
015524 000137  LOOPBX: JMP   @#LOOPC
015526 015056
```

ENTRY POINT FROM OS

```
013400 004737  ENTRY:  JSR   PC, @#M12MC            MASTER CLEAR M1 & M2
013402 015000
013404 012737          MOV   #6Ø, @#FSMCSR
013406 000060                                       INITIALIZE CSR'S
013410 164100
013412 012737          MOV   #2Ø, @#CAMCSR
013414 000020
013416 164110
013420 000412          BR    SIMINT                 SIMULATE AN INTERRUPT
```

INTERRUPT WAIT LOOP

```
013422 105737  IWL:    TSTB  @#TKS                  CHECK FOR TTY CHARACTER
013424 177560
013426 100004          BPL   IWL2                   NO CHARACTER
013430 004737          JSR   PC, @# M12MC           MASTER CLEAR M1 & M2
013432 015500
013434 104400          IN                           READ CHARACTER
013436 000207          RTS   PC                     RETURN TO OS
013440 005227  IWL2:   INC   (PC)+                  INC. INT. TIME LIMIT
013442 000000  ITL:    +Ø
013444 001366          BNE   IWL
```

SIMULATE AN INTERRUPT

```
013446 005046  SIMINT: CLR   -(SP)                  PUSH STATUS WORD
013450 012746          MOV   #IWL, -(SP)            PUSH RETURN POINT
013452 013422
013454 000137          JMP   @#INT                  ENTER INT SERV ROUTINE
013456 016000
```

```
                  .TITLE MAIN  PROGRAM MODEL 1240 OPERATING SYSTEM

                  ;ASSEMBLY PARAMETERS

                  ;MNEMONIC=VALUE  DESCRIPTION

           164000 MBASE1=164000    ;UPROC1 BASE ADDRESS
           164040 MBASE2=164040    ;UPROC2 BASE ADDRESS
           164100 FSMCS=164100      ;FSM1 CONTROL • STATUS
           164100 CAMCS=164100       ;CAMERA CONTROL • STATUS


           000000            .ASECT  ;ERROR + I/O TRAPS
           000004 .=4
    000004 000006            .WORD   .+2
    000006 000002            RTI
           000060 .=60
    000060 000062            .WORD   .+2
    000062 000002            RTI
    000064 000066            .WORD   .+2
    000066 000002            RTI
           000000            .CSECT

                  ;ADDRESSES ASSIGNED TO MICROPROCESSOR

                  ;MNEMONIC  ADDRESS    INPUT          OUTPUT

    000000 164000 MD: MCLR:  MBASE1+00  ;D REGISTER    MASTER CLEAR
    000002 164002 MHALT:     MBASE1+02  ;              HALT
    000004 164004 MS:        MBASE1+04  ;S REGISTER
    000006 164006            MBASE1+06  ;
    000010 164010 MY:        MBASE1+10  ;              DEST CONTROL
    000012 164012 MX:        MBASE1+12  ;              SOURCE CONTROL
    000014 164014 MZ:        MBASE1+14  ;
    000016 164016 MJ: MI:    MBASE1+16  ;OUTPUT FIFO   INPUT FIFO
    000020 164020 MGO:       MBASE1+20  ;              GO/STEP
    000022 164022 MP:        MBASE1+22  ;UPROC STATUS  UPROC CONTROL
    000024 164024            MBASE1+24  ;
    000026 164026 MB:        MBASE1+26  ;              BREAKPOINT
    000030 164030 MA:        MBASE1+30  ;UINST ADDR    UINST ADDR
    000032 164032 MI3:       MBASE1+32  ;UINST 15-0    UINST 15-0
    000034 164034 MI2:       MBASE1+34  ;UINST 31-16   UINST 31-16
    000036 164036 MI1:       MBASE1+36  ;UINST 47-32   UINST 47-32
    000040 000060'MYIA:      IM1        ;MY IMAGE      MY IMAGE
    000042 000062'MXIA:      IM1+2      ;MX IMAGE      MX IMAGE
    000044 000064'MIIA:      IM1+4      ;MI IMAGE      MI IMAGE
    000046 000066'MBIA:      IM1+6      ;MB IMAGE      MB IMAGE
    000050 000070'RFLAG:     IM1+10     ;RUN FLAG      RUN FLAG
    000052 000072'RCNT:      IM1+12     ;RUN COUNT     RUN COUNT
    000054 000074'ROPT:      IM1+14     ;RUN OPTION    RUN OPTION
    000056 000076'LOPT:      IM1+16     ;LIST OPTION   LIST OPTION

    000060 000000 IM1:       +0,0,0,0,0,0,'F,'D
    000062 000000
    000064 000000
```

```
000066 000000
000070 000000
000072 000000
000074 000106
000076 000104
000100 000000 IM2:   +0,0,0,0,0,0,'F,'D
000102 000000
000104 000000
000106 000000
000110 000000
000112 000000
000114 000106
000116 000104



                ;CHARACTERS

         000007 BELL=7
         000015 CR=15
         000012 LF=12
         000030 DELCH='X-100  ;LINE DELETE
         000020 MCLRCH='P-100 ;MASTER CLEAR
         000007 GOCH='G-100   ;GO

                ;GLOBALS

                .GLOBL TYPE,READ,OCTAL,STRING,CRLF
                .GLOBL MA
                .GLOBL TTYBEG,TTYEND,BRANCH
                .GLOBL RETUR.,BUFFER,COLUMN
                .GLOBL UPPER,HTEST.

                ;REGISTERS

         000000 R=%0
         000000 R0=%0
         000001 R1=%1
         000002 R2=%2
         000003 R3=%3
         000004 R4=%4
         000005 R5=%5
         000006 SP=%6
         000007 PC=%7
         177560 TKS=177560

                ;BITS

         000001 BIT0=1
         000002 BIT1=2
         000004 BIT2=4
         000010 BIT3=10
         000020 BIT4=20
         000040 BIT5=40
         000100 BIT6=100
         000200 BIT7=200
```

```
        000400 BIT8=400
        001000 BIT9=1000
        002000 BIT10=2000
        004000 BIT11=4000
        010000 BIT12=10000
        020000 BIT13=20000
        040000 BIT14=40000
        100000 BIT15=100000

               ;FLAGS, COUNTERS AND BUFFERS FOR BOTH
               ;MICROPROCESSORS

000120 000000 PTFLAG: +0
000122 000000 SAVCH:  +0
000124    040 CBEG:    .ASCII  /      /
000125    040
000126    040
000127    040
000130    040
000131    040
000132 000124'CEND:    +CBEG
000134 000000 NBEG:    +0,0,0,0,0
000136 000000
000140 000000
000142 000000
000144 000000
000146 000134'NEND:    +NBEG
000150 000134'PNBEG:   +NBEG


               ;TRAPS

        104420 RETURN=104420
        104422 HTEST=104422
```

```
                ;MAIN PROGRAM MODEL 1240 OPERATING SYSTEM
                ;TAPE 2


                ;BEGINNING OF PROGRAM

000152 012706 BEGIN:  MOV     #1000,SP
       001000
000156 012767'        MOV     #1,TKS
       000001
       177560
000164 000000'        STRING                     ;TYPE OPENING MESSAGE
000166    015         .BYTE   CR
000167    115         .ASCII  /MODEL 1240 OPERATING SYSTEM/
000170    117
000171    104
000172    105
000173    114
000174    040
000175    061
000176    062
000177    064
000200    060
000201    040
000202    117
000203    120
000204    105
000205    122
000206    101
000207    124
000210    111
000211    116
000212    107
000213    040
000214    123
000215    131
000216    123
000217    124
000220    105
000221    115
000222    015         .BYTE   CR,0
000223    000
       000224         .EVEN
000224 005037         CLR     @#MBASE1           ;MASTER CLEAR
       164000
000230 005037         CLR     @#MBASE2           ;BOTH MICROPROCESSORS
       164040
000234 012767'        MOV     #MEMTRP,4          ;SET MEMORY TRAP
       004054
       000004
```

```
                    ;START NEW LINE

000242 105767'LINE:   TSTB     TKS
       177560
000246 100411         BMI      SCAN     ;CHAR READY
000250 005777 TST     @RFLAG
       177574
000254 001406         BEQ      SCAN     ;NOT RUNNING
000256 032777         BIT      #BIT9,@MP  ;HALTED?
       001000
       177536
000264 001766         BEQ      LINE     ;NO
000266 005077         CLR      @RFLAG
       177556

                    ;SCAN LINE

000272 012700'SCAN:   MOV      #NBEG,R
       000134
000276 010067         MOV      R,NEND
       177644
000302 005020 SCAN2:  CLR      (R)+
000304 020027'        CMP      R,#NBEG+10.
       000146
000310 100774         BMI      SCAN2              ;CLEAR NUMBER BUFFER



000312 012767'        MOV      #CBEG,CEND         ;CLEAR COMMAND BUFFER
       000124
       177612
000320 012767'        MOV      #DONOTH,CPOINT  ;DEFAULT COMMAND IS DO NOTHING
       004052
       001610
000326 005067         CLR      SAVCH
       177570

                    ;GET CHARACTER AND BRANCH ACCORDINGLY

000332 016700 CHAR:   MOV      SAVCH,R
       177564
000336 001001         BNE      CHAR2              ;USE SAVED CHAR IF NONZERO
000340 000000'        READ
000342 005067 CHAR2:  CLR      SAVCH
       177554
000346 000000'        BRANCH
000350    040         .BYTE     ,SPACE-.
000351    035
000352    015         .BYTE    CR,ELINE-.
000353    037
000354    030         .BYTE    DELCH,DLINE-.
000355    041
000356    007         .BYTE    GOCH,GO-.
000357    045
000360    020         .BYTE    MCLRCH,MASTER-.
000361    275
000362    073         .BYTE    ';,COMMEN-.
000363    003
000364    000         .BYTE    0,NUMBER-.
000365    321
```

;SERVICE ROUTINES FOR ONE-CHARACTER ITEMS

```
000366 000000'COMMEN: TYPE                      ;COMMENT EXTENDS TO
000370 000000'        READ
000372 000000'        BRANCH
000374    015         .BYTE   CR,ELINE-.        ;CARRIAGE RETURN,
000375    015
000376    030         .BYTE   DELCH,DLINE-.     ;LINE DELETE
000377    017
000400    020         .BYTE   MCLRCH,MASTER-.   ;OR MASTER CLEAR
000401    255
000402    000         .BYTE   0,1
000403    001
000404 000710         BR      COMMEN

000406 000000'SPACE:  TYPE                      ;ECHO SPACE BUT IGNORE IT
000410 000700         BR      CHAR

000412 016707 ELINE:  MOV     CPOINT,PC         ;END OF LINE
       001520
                                                ;GO TO CURRENT SERVICE ROUTINE

000416 000000'DLINE:  STRING                    ;DELETE LINE



000420    130         .BYTE   'X,0              ;ECHO X
000421    000
000422 104420         RETURN
000424 022767'GO:     CMP     #BUFFER,COLUMN
       000000
       000000
000432 001125         BNE     NUMBER  ;NOT FIRST CHAR
000434 104422         NTEST
000436 017700         MOV     @LOPT,R
       177414
000442 067700         ADD     @ROPT,R
       177406
000446 020027         CMP     R,#'D+'F
       000212
000452 001377         BNE     GO2
000454 042777         BIC     #BIT5,@MP         ;LIST DISABLED, FREE MODE
       000040
       177340
000462 005277         INC     @RFLAG
       177362
000466 005077         CLR     @MGO
       177326
000472 000000'        STRING
000474    107         .ASCII  /GO/
000475    117
000476    000         .BYTE   0
       000500         .EVEN
000500 104420         RETURN
```

```
000502 052777 GO2:     BIS     #BIT5,@MP ;OTHER CASES, USE STEP
       000040
       177312
000510 020027          CMP     R,#'D+'S
       000227
000514 001005          BNE     GO3
000516 004767          JSR     PC,STEP    ;LIST DISABLED, STEP MODE
       000120
000522 005000          CLR     R
000524 104402          TRAP    2   ;TWITCH
000526 104420          RETURN
000530 020027 GO3:     CMP     R,#'E+'F
       000213
000534 001012          BNE     GO4
000536 004767          JSR     PC,LSTEP          ;LIST ENABLED, FREE MODE
       000026
000542 032777          BIT     #BIT7,@MP         ;HALTED?
       000200
       177252
000550 001003          NOP                       ;YES
000552 004567          JSR     R5,ABORT
       003354
000556 000764          BR      GO3+6
000560 104420 GOX:     RETURN
000562 004767 GO4:     JSR     PC,LSTEP          ;LIST ENABLED, STEP MODE
       000002
000566 104420          RETURN




000570 000000'LSTEP:   STRING
000572    055          .BYTE   '-,0
000573    000
000574 000000'         OCTAL
000576 000030'         +MA,4
000600 000004
000602 004767          JSR     PC,TMINST
       002346
000606 004767          JSR     PC,STEP
       000030
000612 000000'         OCTAL
000614 000000'         +MD,4
000616 000004
000620 012700          MOV     #'W,R
       000127
000624 032777          BIT     #BIT10+BIT11,@MP
       006000
       177170
000632 001401          BEQ     .+4
000634 000000'         TYPE
000636 000000'         CRLF
000640 000207          RTS     PC
```

```
000642 005077 STEP:   CLR     @MGO
       177152
000646 000240         NOP
000650 005077         CLR     @MHALT
       177126
000654 000207         RTS     PC

000656 005077 MASTER: CLR     @MCLR              ;MASTER CLEAR
       177116
000662 000000'        STRING
000664    115         .ASCII  /MASTER CLEAR /
000665    101
000666    123
000667    124
000670    105
000671    122
000672    040
000673    103
000674    114
000675    105
000676    101
000677    122
000700    040
000701    000         .BYTE   0
       000702         .EVEN
000702 000167         JMP     TMP
       003064


              ;SCAN FOR NUMBER

              NUMBER:
000706 020027         CMP     R,#'0
       000060


000712 100457         BMI     COMMAN             ;GO TO COMMAN(D) IF
000714 020027         CMP     R,#'8              ;CHARACTER IS NOT
       000070
000720 100054         BPL     COMMAN             ;AN OCTAL DIGIT

000722 022767'        CMP     #NBEG+10.,NEND
       000146
       177216
000730 100002         BPL     NUMT
000732 000167         JMP     ERROR              ;TOO MANY NUMBERS
       003106
```

```
000736 005003 NUMT:  CLR    R3              ;R3 HOLDS NUMBER
000740 006303 NUM2:  ASL    R3
000742 006303        ASL    R3
000744 006303        ASL    R3              ;MULTIPLY BY 8
000746 060003        ADD    R,R3
000750 162703        SUB    #'0,R3          ;AND ADD DIGIT
       000060
000754 000000'       TYPE                   ;ECHO DIGIT
000756 000000'NUM3:  READ                   ;GET NEXT DIGIT
000760 000000'       BRANCH
000762    060        .BYTE  '0,NUM5-.
000763    033
000764    061        .BYTE  '1,NUM5-.
000765    031
000766    062        .BYTE  '2,NUM5-.
000767    027
000770    063        .BYTE  '3,NUM5-.
000771    025
000772    064        .BYTE  '4,NUM5-.
000773    023
000774    065        .BYTE  '5,NUM5-.
000775    021
000776    066        .BYTE  '6,NUM5-.
000777    017
001000    067        .BYTE  '7,NUM5-.
001001    015
001002    040        .BYTE  ' ,NUM6-.
001003    015
001004    015        .BYTE  CR,NUM6-.
001005    013
001006    030        .BYTE  DELCH,NUM6-.
001007    011
001010    020        .BYTE  MCLRCH,NUM6-.
001011    007
001012    073        .BYTE  ';,NUM6-.
001013    005
001014    000        .BYTE  0,NUMERR-.
001015    027

001016 000750 NUM5:  BR     NUM2

001020 010377 NUM6:  MOV    R3,@NEND        ;SAVE NUMBER
       177122
001024 062767        ADD    #2,NEND
       000002
       177114
```

```
001032 110067          MOVB    R,SAVCH          ;SAVE CHARACTER
       177064
001036 012700          MOV     #'#,R            ;# MEANS NUMBER
       000043
001042 000403          BR      COMMAN

001044 000000'NUMERR:  STRING
001046    007          .BYTE   BELL,0           ;RING BELL
001047    000
001050 000742          BR      NUM3             ;AND KEEP SCANNING

               ;INSERT CHARACTER INTO COMMAND BUFFER

001052 022767'COMMAN:  CMP     #CBEG+6,CEND
       000132
       177052
001060 100002          BPL     COM2
001062 000167          JMP     ERROR            ;TOO MANY ITEMS
       002756
001066 110077 COM2:    MOVB    R,@CEND
       177040
001072 005267          INC     CEND
       177034

               ;LOOK UP IN COMMAND TABLE

001076 012703'         MOV     #CTAB,R3
       001204

001102 012701'LOOK:    MOV     #CBEG,R1         ;R1 POINTS TO COMMAND BUFFER
       000124
001106 010302          MOV     R3,R2            ;R2 POINTS TO CURRENT TABLE ENTRY
001110 105712          TSTB    (R2)
001112 001422          BEQ     LOOK4            ;END OF TABLE, NOT FOUND

001114 020167 LOOK2:   CMP     R1,CEND          ;CHECK FOR END OF COMMAND
       177012
001120 100005          BPL     LOOK3            ;END - MATCH FOUND
001122 122122          CMPB    (R1)+,(R2)+      ;COMPARE CHARACTERS
001124 001773          BEQ     LOOK2            ;IF EQUAL, KEEP LOOKING
001126 062703          ADD     #8.,R3
       000010
001132 000763          BR      LOOK             ;IF NOT, GO TO NEXT ENTRY

001134 062703 LOOK3:   ADD     #6,R3
       000006
001140 011367          MOV     (R3),CPOINT      ;CHANGE COMMAND POINTER
       000772
001144 020027          CMP     R,#'#
       000043
001150 001401          BEQ     LOOKX
001152 000000'         TYPE                     ;ECHO NON-NUMBER CHARACTER
001154 000167 LOOKX:   JMP     CHAR             ;GET NEXT CHARACTER
       177152
```

```
001160 005367 LOOK4:  DEC     CEND              ;REMOVE BAD CHAR FROM BUFFER
       176746
001164 020027         CMP     R,#'#
       000043
001170 001002         BNE     LOOK5
001172 000167         JMP     ERROR             ;BAD NUMBER IS HOPELESS
       002646
001176 000000'LOOK5:  STRING
001200    007         .BYTE   BELL,0            ;RING BELL
001201    000
001202 000764         BR      LOOKX             ;GET NEXT CHARACTER
                      .EOT

                ;MAIN PROGRAM MODEL 1240 OPERATING SYSTEM
                ;TAPE 3

                ;COMMAND TABLE
                CTAB:
                ;       SOURCE  ENTRY   DESCRIPTION
                ;        CODE   POINT

001204    124 .ASCII  /T      /         ;TYPE MICROLOCATION +1
001205    040
001206    040
001207    040
001210    040
001211    040
001212 002770'                +T
001214    124 .ASCII  /TA     /         ;TYPE ADDRESS OF CURRENT UINSTR
001215    101
001216    040
001217    040
001220    040
001221    040
001222 002662'                +TA
001224    124 .ASCII  /T#     /         ;TYPE UINSTR AT ADDRESS #
001225    043
001226    040
001227    040
001230    040
001231    040
001232 002674'                +TN
001234    124 .ASCII  /T##    /         ;TYPE UINSTS AT LOCS # TO #
001235    043
001236    043
001237    040
001240    040
001241    040
001242 003046'                +TNN
001244    124 .ASCII  /TU     /         ;TYPE UNIBUS LOCATION + 2
001245    125
001246    040
001247    040
001250    040
001251    040
001252 002156'                +TU
001254    124 .ASCII  /TU#    /         ;TYPE UNIBUS LOCATION #
```

```
001255     125
001256     043
001257     040
001260     040
001261     040
001262 002140'               +TUN
001264     124 .ASCII  /TU##  /           ;TYPE UNIBUS LOCATIONS # TO #
001265     125
001266     043
001267     043
001270     040
001271     040
001272 002202'               +TUNN
001274     103 .ASCII  /CU##  /           ;CHANGE UNIBUS LOCATION # TO #
001275     125
001276     043
001277     043
001300     040
001301     040
001302 002272'               +CUNN
001304     117 .ASCII  /OSOP  /           ;OP SYS OVERSTORE PROTECT
001305     123
001306     117
001307     120
001310     040
001311     040
001312 002340'               +OSOP
001314     117 .ASCII  /OSOPD /           ;OP SYS OVERSTORE PROTECT DISABLE
001315     123
001316     117
001317     120
001320     104
001321     040
001322 002332'               +OSOPD
001324     105 .ASCII  /EXI   /
001325     130
001326     111
001327     040
001330     040
001331     040
001332 004044'               +ERROR
001334     105 .ASCII  /EXIT  /           ;EXIT FROM OPERATING SYSTEM
001335     130
001336     111
001337     124
001340     040
001341     040
001342 004120'               +EXIT
001344     120 .ASCII  /PT    /           ;END PAPER TAPE COMMANDS
001345     124
001346     040
001347     040
001350     040
001351     040
001352 004036'               +PTE
001354     120 .ASCII  /PTB   /           ;BEGIN PAPER TAPE COMMANDS
```

```
001355     124
001356     102
001357     040
001360     040
001361     040
001362 004026'                  +PTB
001364     102 .ASCII  /BD    /             ;BREAKPOINT DISABLE
001365     104
001366     040
001367     040
001370     040
001371     040
001372 002350'                  +BD
001374     102 .ASCII  /BE    /             ;BREAKPOINT ENABLE
001375     105
001376     040
001377     040
001400     040
001401     040
001402 002362'                  +BE
001404     112 .ASCII  /JZ    /             ;JUMP TO ZERO
001405     132
001406     040
001407     040
001410     040
001411     040
001412 002374'                  +JZ
001414     112 .ASCII  /JB    /             ;JUMP TO BREAKPOINT
001415     102
001416     040
001417     040
001420     040
001421     040
001422 002406'                  +JB
001424     114 .ASCII  /L     /             ;TYPE LISTING STATUS
001425     040
001426     040
001427     040
001430     040
001431     040
001432 002432'                  +LSTAT
001434     114 .ASCII  /LD    /             ;NO LISTING
001435     104
001436     040
001437     040
001440     040
001441     040
001442 002420'                  +LD
001444     114 .ASCII  /LE    /             ;LIST  ENABLE
001445     105
001446     040
001447     040
001450     040
001451     040
001452 002420'                  +LE
001454     122 .ASCII  /R     /             ;TYPE RUN STATUS
```

```
001455     040
001456     040
001457     040
001460     040
001461     040
001462 002454'                      +RSTAT
001464     122 .ASCII  /RS    /           ;STEP MODE
001465     123
001466     040
001467     040
001470     040
001471     040
001472 002442'                      +RS
001474     122 .ASCII  /RF    /           ;FREE MODE
001475     106
001476     040
001477     040
001500     040
001501     040
001502 002442'                      +RF
001504     124 .ASCII  /TI    /           ;TYPE INPUT FIFO
001505     111
001506     040
001507     040
001510     040
001511     040
001512 003200'                      +TI
001514     124 .ASCII  /TJ    /           ;TYPE OUTPUT FIFO (TOP)
001515     112
001516     040
001517     040
001520     040
001521     040
001522 002472'                      +TJ
001524     124 .ASCII  /TD    /           ;TYPE D REGISTER
001525     104
001526     040
001527     040
001530     040
001531     040
001532 003216'                      +TD
001534     124 .ASCII  /TS    /           ;TYPE S REGISTER
001535     123
001536     040
001537     040
001540     040
001541     040
001542 003230'                      +TS
001544     124 .ASCII  /TP    /           ;TYPE MICROPROCESSOR STATUS
001545     120
001546     040
001547     040
001550     040
001551     040
001552 003242'                      +TP
001554     124 .ASCII  /TB    /           ;TYPE BREAKPOINT
```

```
001555    102
001556    040
001557    040
001560    040
001561    040
001562 003252'           +TR
001564    103 .ASCII  /CA#   /          ;CHANGE MICROPROCESSOR ADDRESS
001565    101
001566    043
001567    040
001570    040
001571    040
001572 003302'           +CAN
001574    103 .ASCII  /C#### /          ;CHANGE MICROINSTRUCTION
001575    043
001576    043
001577    043
001600    043
001601    040
001602 003332'           +CNNNN
001604    103 .ASCII  /CI#   /          ;CHANGE INPUT FIFO
001605    111
001606    043
001607    040
001610    040
001611    040
001612 003472'           +CIN
001614    103 .ASCII  /CP#   /          ;CHANGE MICROPROCESSOR CONTROL
001615    120
001616    043
001617    040
001620    040
001621    040
001622 003520'           +CPN
001624    103 .ASCII  /CB#   /          ;CHANGE BREAKPOINT
001625    102
001626    043
001627    040
001630    040
001631    040
001632 003536'           +CBN
001634    103 .ASCII  /CX#   /          ;CHANGE SOURCE CONTROL
001635    130
001636    043
001637    040
001640    040
001641    040
001642 003556'           +CXN
001644    103 .ASCII  /CY#   /          ;CHANGE DESTINATION CONTROL
001645    131
001646    043
001647    040
001650    040
001651    040
001652 003576'           +CYN
001654    103 .ASCII  /CM### /          ;CHANGE CURRENT MICROINSTRUCTION
```

```
001655     115
001656     043
001657     043
001660     043
001661     040
001662 003616'                  +CMNNN
001664     103 .ASCII  /CIMP  /        ;INPUT MODE PIXEL
001665     111
001666     115
001667     120
001670     040
001671     040
001672 002526'                  +CIMP
001674     103 .ASCII  /CJMP  /        ;OUTPUT MODE PIXEL
001675     112
001676     115
001677     120
001700     040
001701     040
001702 002534'                  +CJMP
001704     103 .ASCII  /CIMW  /        ;INPUT MODE WORD
001705     111
001706     115
001707     127
001710     040
001711     040
001712 002550'                  +CIMW
001714     103 .ASCII  /CJMW  /        ;OUTPUT MODE WORD
001715     112
001716     115
001717     127
001720     040
001721     040
001722 002556'                  +CJMW
001724     124 .ASCII  /TM    /        ;TYPE CURRENT MICROINSTRUCTION
001725     115
001726     040
001727     040
001730     040
001731     040
001732 002736'                  +TM
001734     115 .ASCII  /M     /        ;TYPE MICROPROCESSOR NUMBER
001735     040
001736     040
001737     040
001740     040
001741     040
001742 003772'                  +TMP
001744     115 .ASCII  /M#    /        ;CHANGE MICROPROCESSOR
001745     043
001746     040
001747     040
001750     040
001751     040
001752 003672'                  +CMP
001754     124 .ASCII  /TX    /        ;TYPE SOURCE CONTROL
```

```
001755     130
001756     040
001757     040
001760     040
001761     040
001762 003262'            +TX
001764     124 .ASCII  /TY    /           ;TYPE DESTINATION CONTROL
001765     131
001766     040
001767     040
001770     040
001771     040
001772 003272'            +TY
001774     124 .ASCII  /TIM   /           ;TYPE INPUT FIFO MODE
001775     111
001776     115
001777     040
002000     040
002001     040
002002 002572'            +TIM
002004     124 .ASCII  /TJM   /           ;TYPE OUTPUT FIFO MODE
002005     112
002006     115
002007     040
002010     040
002011     040
002012 002600'            +TJM
002014     110 .ASCII  /HALT  /           ;HALT MICROPROCESSOR
002015     101
002016     114
002017     124
002020     040
002021     040
002022 002464'            +HALT
002024     130 .ASCII  /XQT## /           ;INVALID
002025     121
002026     124
002027     043
002030     043
002031     040
002032 004044'            +ERROR
002034     130 .ASCII  /XQT###/           ;EXECUTE MICROINSTRUCTION
002035     121
002036     124
002037     043
002040     043
002041     043
002042 002626'            +XQTNNN
002044     114 .ASCII  /LM    /           ;INVALID
002045     115
002046     040
002047     040
002050     040
002051     040
002052 004044'            +ERROR
002054     114 .ASCII  /LMP#  /           ;LOAD MICROPROGRAM FROM PTAPE
```

```
002055    115
002056    120
002057    043
002060    040
002061    040
002062 004014'              +LMPN
002064    104 .ASCII  /DUMP  /         ;DUMP UPROC
002065    125
002066    115
002067    120
002070    040
002071    040
002072 004444'              +DUMP
002074    114 .ASCII  /LQT   /         ;LOAD QUANTIZER
002075    121
002076    124
002077    040
002100    040
002101    040
002102 004250'              +LQT
002104    123 .ASCII  /SNAP  /         ;SNAP A PICTURE
002105    116
002106    101
002107    120
002110    040
002111    040
002112 004206'              +SNAP
002114    112 .ASCII  /JSR   /         ;INVALID
002115    123
002116    122
002117    120
002120    103
002121    040
002122 004044'              +ERROR
002124    112 .ASCII  /JSR###/         ;JSR PC,#
002125    123
002126    122
002127    120
002130    103
002131    043
002132 004434'              +JSRPC
002134 000000 +0

002136 004052'CPOINT: +DONOTH
                .EOT


          ;MAIN PROGRAM MODEL 1240 OPERATING SYSTEM
          ;TAPE 4

          ;SERVICE ROUTINES FOR UNIBUS COMMANDS

002140 016737 TUN:    MOV     NBEG,@(PC)+
       175770
002144 002162'PUHOLD: +UHOLL
002146 000000'        OCTAL
002150 000134'        NBEG,6
002152 000006
```

```
002154 104420          RETURN

002156 062727 TO:      ADD     #2,(PC)+
       000002
002162 000000 UHOLD:   +0
002164 000000'         OCTAL
002166 002144'         PUHOLD,6
002170 000006
002172 000000'OCTAL
002174 002162'         UHOLD,6
002176 000006
002200 104420          RETURN

002202 000000'TUNN:    CRLF
002204 000000'         OCTAL
002206 000150'         PNBEG,6
002210 000006
002212 000000'         STRING
002214    040          .ASCII  / /
002215    040
002216    000          .BYTE   0
       002220          .EVEN
002220 012703          MOV     #7,R3
       000007

002224 000000'TUNN2:   OCTAL
002226 000134'         NBEG,6
002230 000006
002232 016767          MOV     NBEG,UHOLD
       175676
       177722
002240 062767          ADD     #2,NBEG
       000002
       175666
002246 026767          CMP     NBEG,NBEG+2
       175662
       175662
002254 101005          BHI     TUNNX
002256 005303          DEC     R3
002260 100361          BPL     TUNN2
002262 004567          JSR     R5,ABORT
       001644
002266 000745          BR      TUNN
002270 104420 TUNNX:   RETURN

002272 005767 CUNN:    TST     OSOPF              ;CHECK WHETHER OVERSTORE
       000046
002276 001404          BEQ     CUNN2              ;PROTECT IS IN EFFECT
002300 026727'         CMP     NBEG,#UPPER        ;UPPER LIMIT OF OS
       175630
       000000
002306 103407          BLO     CUNNE              ;TREAT LIKE MEMORY TRAP
002310 016777 CUNN2:   MOV     NBEG+2,@NBEG       ;MAKE THE CHANGE
       175622
       175616
002316 016767          MOV     NBEG,UHOLD
       175612
       177636
```

```
002324 104420          RETURN
002326 000167 CUNNE:   JMP     MEMTRP
       001522

002332 005067 OSOPD:   CLR     OSOPF
       000006
002336 104420          RETURN
002340 012727 OSOP:    MOV     #1,(PC)+
       000001
002344 000001 OSOPF:   +1
002346 104420          RETURN


                       ;BREAKPOINT ENABLE/DISABLE

002350 104422 BD:      HTEST
002352 042777          BIC     #BIT3,@MP
       000010
       175442
002360 104420          RETURN
002362 104422 BE:      HTEST
002364 052777          BIS     #BIT3,@MP
       000010
       175430
002372 104420          RETURN

                       ;JUMP OPTIONS

002374 104422 JZ:      HTEST
002376 042777          BIC     #BIT4,@MP
       000020
       175416
002404 104420          RETURN
002406 104422 JB:      HTEST
002410 052777          BIS     #BIT4,@MP
       000020
       175404
002416 104420          RETURN

                       ;LIST OPTIONS

002420 104422 LE:LD:   HTEST
002422 116777          MOVB    CREG+1,@LOPT
       175477
       175426
002430 104420          RETURN
002432 017700 LSTAT:   MOV     @LOPT,R
       175420
002436 000000'         TYPE
002440 104420          RETURN
                ;RUN OPTIONS

002442 104422 RS:RF:   HTEST
002444 116777          MOVB    CREG+1,@ROPT
       175455
       175402
```

```
002452 104420           RETURN
002454 017700 RSTAT:    MOV      @ROPT,R
       175374
002460 000000'          TYPE
002462 104420           RETURN

002464 010077 HALT:     MOV      R,@MHALT
       175312
002470 104420           RETURN
002472 032777 TJ:       BIT      #BIT14,@MP
       040000
       175322
002500 001406           BEQ      TJ1
002502 000000'          STRING
002504    040           .ASCII   / EMPTY/
002505    105
002506    115
002507    120
002510    124
002511    131
002512    000           .BYTE    0
       002514           .EVEN
002514 104420           RETURN
002516 000000'TJ1:      OCTAL
002520 000016'          MJ,6
002522 000006
002524 104420           RETURN
       ;
              ;INPUT AND OUTPUT FIFO MODES

002526 012702 CIMP:     MOV      #BIT0,R2
       000001
002532 000402           BR       CJMP2
002534 012702 CJMP:     MOV      #BIT1,R2
       000002
002540 104422 CJMP2:    HTEST
002542 050277           BIS      R2,@MP
       175254
002546 104420           RETURN
002550 012702 CIMW:     MOV      #BIT0,R2
       000001
002554 000402           BR       CJMW2
002556 012702 CJMW:     MOV      #BIT1,R2
       000002
002562 104422 CJMW2:    HTEST
002564 040277           BIC      R2,@MP
       175232
002570 104420           RETURN
002572 012702 TIM:      MOV      #BIT0,R2
       000001
002576 000402           BR       TJM2
002600 012702 TJM:      MOV      #BIT1,R2
       000002
002604 012700 TJM2:     MOV      #'W,R
       000127
002610 030277           BIT      R2,@MP
       175206
```

```
002614 001402          BEQ     TJM3
002616 012700          MOV     #'P,R
       000120
002622 000000'TJM3:    TYPE
002624 104420          RETURN

002626 104422 XCTNNN:  HTEST
002630 012700'         MOV     #NBEG,R0
       000104
002634 012701'         MOV     #XQT2,R1
       002652
002640 012021          MOV     (R0)+,(R1)+
002642 012021          MOV     (R0)+,(R1)+
002644 011011          MOV     (R0),(R1)
002646 004567          JSR     R5,XMI
       002054
002652 000000 XQT2:    +0,0,0
002654 000000
002656 000000
002660 104420          RETURN

                       ;TYPE MICROADDRESS AND MICROINSTRUCTION

002662 104422 TA:      HTEST
002664 000000'         OCTAL
002666 000030'         MA,4
002670 000004
002672 104420          RETURN
002674 016737 TN:      MOV     NBEG,@(PC)+
       175234
002700 002776'PMHOLD:  +MHOLD
002702 104422          HTEST
002704 017746          MOV     @MA,-(SP)
       175120
002710 016777          MOV     NBEG,@MA
       175220
       175112
002716 004767          JSR     PC,TMINST
       000232
002722 012677          MOV     (SP)+,@MA
       175102
002726 042777          BIC     #176000,@MA
       176000
       175074
002734 104420          RETURN
002736 104422 TM:      HTEST
002740 000000'         OCTAL
002742 000030'         MA,4
002744 000004
002746 017767          MOV     @MA,MHOLD
       175056
       000022
002754 042767          BIC     #176000,MHOLD
       176000
       000014
```

```
002762 004767          JSR     PC,[illegible]
       000166
002766 104420          RETURN
002770 104422 T:       HTEST
002772 062727          ADD     #1,(PC)+
       000001
002776 000000 MHOLD:   0
003000 000000'         OCTAL
003002 002776'         MHOLD,4
003004 000004
003006 017746          MOV     @MA,-(SP)
       175010
003012 016777          MOV     MHOLD,@MA
       177760
       175010
003020 042777          BIC     #176000,@MA
       176000
       175002
003026 004767          JSR     PC,TMINST
       000122
003032 012677          MOV     (SP)+,@MA
       174772
003036 042777          BIC     #176000,@MA
       176000
       174764
003044 104420          RETURN
003046 104422 TNN:     HTEST
003050 017746          MOV     @MA,-(SP)
       174754
003054 016777          MOV     NBEG,@MA
       175054
       174746
003062 000000'TNN2:    CRLF
003064 032777          BIT     #7,@MA
       000007
       174736
003072 001001          BNE     TNN3
003074 000000'         CRLF
003076 000000'TNN3:    OCTAL
003100 000030'         MA,4
003102 000004
003104 017767          MOV     @MA,MHOLD
       174720
       177664
003112 004767          JSR     PC,TMINST
       000036
003116 027767          CMP     @MA,NBEG+2
       174706
       175012
003124 100005          BPL     TNNX
003126 005277          INC     @MA
       174676
003132 004567          JSR     R5,ABORT
       000774
003136 000751          BR      TNN2
003140 012677 TNNX:    MOV     (SP)+,@MA
       174664
```

```
003144 042777          BIC     #176000,@MA
       176000
       174656
003152 104420          RETURN

003154 000000'TMINST:  OCTAL             ;TYPE CURRENT MICROINSTRUCTION
003156 000036'         MI1,6
003160 000006
003162 000000'         OCTAL
003164 000034'         MI2,6
003166 000006
003170 000000'         OCTAL
003172 000032'         MI3,6
003174 000006
003176 000207          RTS     PC

                       ;TYPE VARIOUS MICROPROCESSOR REGISTERS

003200 000000'TI:      OCTAL
003202 000044'         MIIA,4
003204 000004
003206 000000'TAST:    STRING
003210    040          .ASCII  / */
003211    052
003212    000          .BYTE   0
       003214          .EVEN
003214 104420          RETURN
003216 104422 TD:      HTEST
003220 000000'         OCTAL
003222 000000'         MD,4
003224 000004
003226 104420          RETURN
003230 104422 TS:      HTEST
003232 000000'         OCTAL
003234 000004'         MS,4
003236 000004
003240 104420          RETURN
003242 000000'TP:      OCTAL
003244 000022'         MP,6
003246 000006
003250 104420          RETURN
003252 000000'TB:      OCTAL
003254 000046'         MBIA,4
003256 000004
003260 000752          BR      TAST
003262 000000'TX:      OCTAL
003264 000042'         MXIA,6
003266 000006
003270 000746          BR      TAST
003272 000000'TY:      OCTAL
003274 000040'         MYIA,6
003276 000006
003300 000742          BR      TAST

                       ;CHANGE VARIOUS MICROPROC REGISTERS
```

```
003302 104422 CAN:      HTEST
003304 016777           MOV      NBEG,@MA
       174624
       174516
003312 017746           MOV      @MA,-(SP)
       174512
003316 042716           BIC      #176000,(SP)
       176000
003322 026726           CMP      NBEG,(SP)+
       174606
003326 001045           BNE      CERR
003330 104420           RETURN

003332 104422 CNNNN:    HTEST
003334 017746           MOV      @MA,-(SP)
       174470
003340 012700'          MOV      #NBEG,R
       000134
003344 011067           MOV      (R),MHOLD
       177426
003350 011077           MOV      (R),@MA
       174454
003354 017746           MOV      @MA,-(SP)
       174450
003360 042716           BIC      #176000,(SP)
       176000
003364 022026           CMP      (R)+,(SP)+
003366 001025           BNE      CERR
003370 011077           MOV      (R),@MI1
       174442
003374 022077           CMP      (R)+,@MI1
       174436
003400 001020           BNE      CERR
003402 011077           MOV      (R),@MI2
       174426
003406 022077           CMP      (R)+,@MI2
       174422
003412 001013           BNE      CERR
003414 011077           MOV      (R),@MI3
       174412
003420 022077           CMP      (R)+,@MI3
       174406
003424 001006           BNE      CERR
003426 011677           MOV      (SP),@MA
       174376
003432 022677           CMP      (SP)+,@MA
       174372
003436 001001           NOP
003440 104420           RETURN
003442 000000'CERR:     STRING
003444    105           .ASCII   /ERROR IN READ BACK/
003445    122
003446    122
003447    117
003450    122
003451    040
```

```
003452    111
003453    116
003454    040
003455    122
003456    105
003457    101
003460    104
003461    040
003462    102
003463    101
003464    103
003465    113
003466    007          .BYTE   BELL,0
003467    000
       003470          .EVEN
003470 104420          RETURN

003472 016777 CIN:     MOV     NBEG,@MIIA
       174436
       174344
003500 017777          MOV     @MIIA,@MI
       174340
       174310
003506 000000'         STRING
003510    040          .ASCII  / PUSH/
003511    120
003512    125
003513    123
003514    110
003515    000          .BYTE   0
       003516          .EVEN
003516 104420          RETURN
003520 016777 CPN:     MOV     NBEG,@MP
       174410
       174274
003526 017777          MOV     @MP,@MP
       174270
       174266
003534 104420          RETURN
003536 104422 CBN:     HTEST
003540 016777          MOV     NBEG,@MBIA
       174370
       174300
003546 017777          MOV     @MBIA,@MB
       174274
       174252
003554 104420          RETURN
003556 104422 CXN:     HTEST
003560 016777          MOV     NBEG,@MXIA
       174350
       174254
003566 017777          MOV     @MXIA,@MX
       174250
       174216
003574 104420          RETURN
003576 104422 CYN:     HTEST
```

```
003600 016777          MOV     NBEG,@MYIA
       174330
       174232
003606 017777          MOV     @MYIA,@MY
       174226
       174174
003614 104420          RETURN
003616 104422 CMNNN:   RTEST
003620 017767          MOV     @MA,MHOLD
       174204
       177150
003626 012700'         MOV     #NBEG,R
       000134
003632 011077          MOV     (R),@MI1
       174200
003636 022077          CMP     (R)+,@MI1
       174174
003642 001277          BNE     CERR
003644 011077          MOV     (R),@MI2
       174164
003650 022077          CMP     (R)+,@MI2
       174160
003654 001272          BNE     CERR
003656 011077          MOV     (R),@MI3
       174150
003662 022077          CMP     (R)+,@MI3
       174144
003666 001265          BNE     CERR
003670 104420          RETURN

                       ;MICROPROCESSOR SELECTION

003672 116700 CMP:     MOVB    NBEG,R
       174236
003676 005300          DEC     R
003700 046700'         BIC     #177776,R
       177776
003704 006300          ASL     R
003706 016001'         MOV     CMPTAB(R),R1
       003762
003712 005711          TST     (R1)   ;CHECK TO SEE IF MP EXISTS
003714 012702'         MOV     #MD,R2
       000000
003720 010122 CMPL:    MOV     R1,(R2)+
003722 062701          ADD     #2,R1
       000002
003726 020227'         CMP     R2,#MI1+2
       000040
003732 100772          BMI     CMPL
003734 016001'         MOV     CMPTAB+4(R),R1
       003766
003740 012702'         MOV     #MYIA,R2
       000040
003744 010122 CMPL2:   MOV     R1,(R2)+
003746 062701          ADD     #2,R1
       000002
```

```
003752 020227'          CMP     R2,#LOPT+2
       000060
003756 100772           BMI     CMPL2
003760 104420           RETURN
003762 164000 CMPTAB:  +MBASE1
003764 164040          +MBASE2
003766 000060'         +IM1
003770 000100'         +IM2

003772 012700 TMP:      MOV     #'1,R
       000061
003776 026727           CMP     MD,#MBASE1
       173776
       164000
004004 001401           BEQ     TMP2
004006 005200           INC     R
004010 000000'TMP2:     TYPE
004012 104420           RETURN

                        ;COMMANDS THAT CONTROL MICROPROCESSOR


                        .GLOBL  LOAD
004014 016705 LMPN:     MOV     NBEG,R5
       174114
004020 004767'          JSR     PC,LOAD
       000000
004024 104420           RETURN



                        ;INTERNAL OPERATING SYSTEM COMMANDS

004026 012767 PTB:      MOV     #1,PTFLAG
       000001
       174064
004034 104420           RETURN
004036 005067 PTE:      CLR     PTFLAG
       174056
004042 104420           RETURN

004044 000000'ERROR:    STRING
004046    077           .BYTE   '?,BELL,0
004047    007
004050    000
       004052           .EVEN
004052 104420 DONOTH:   RETURN

004054 000000'MEMTRP:   STRING
004056    040           .ASCII  / BAD ADDRESS/
004057    102
004060    101
004061    104
004062    040
004063    101
004064    104
```

```
004065    104
004066    122
004067    105
004070    123
004071    123
004072    007           .BYTE   BELL,0
004073    000
       004074           .EVEN

004074 012706 RETUR.: MOV     #1000,SP
       001000
004100 022767'          CMP     #BUFFER,COLUMN
       000000
       000000
004106 001401           BEQ     RET2
004110 000000'          CRLF
004112 012705'RET2:     JMP     LINE
       000242
004116 000205

004120 000000'EXIT:     CRLF
004122 005067'          CLR     77476
       077476
004126 000167'          JMP     77476
       077476

                        .EOT

              ;MAIN PROGRAM 1240 OPERATING SYSTEM
              ;TAPE 5

004132 105737 ABORT:  TSTB    @#TKS
       177560
004136 100004           BPL     AB2  ;NO CHAR
004140 005767           TST     PTFLAG
       173754
004144 001001           BNE     AB2     ;READING PTAPE
004146 005725           TST     (R5)+ ;ERROR RETURN
004150 000205 AB2:      RTS     R5

004152 005767 HTEST.: TST     @HFLAG
       173672
004156 001005           BNE     HT2     ;RUNNING
004160 032777           BIT     #BIT9,@MP
       001000
       173634
004166 001401           BEQ     HT2     ;NOT HALTED
004170 000205           RTS     R5      ;NOT RUNNING
004172 000000'HT2:      STRING
004174    122           .ASCII  /RUNNING/
004175    125
004176    116
004177    116
004200    111
004201    116
004202    107
004203    000           .BYTE   0
```

```
        004204          .EVEN
004204 104420           RETURN

004206 012700 SNAP:     MOV     #FSMCS,R
       164100
004212 005010           CLR     (R)
004214 005060           CLR     10(R)
       000010
004220 012710           MOV     #61,(R)
       000061
004224 012760           MOV     #1,10(R)
       000001
       000010
004232 000240           NOP
004234 000240           NOP
004236 000240           NOP
004240 032710 SNAP2:    BIT     #BIT0,(R)
       000001
004244 001375           BNE     SNAP2
004246 104420           RETURN

              ;LOAD QUANTIZER

004250 104422 LQT:      HTEST
004252 012701           MOV     #QIMAGE,R1       ;QUANT TABLE IMAGE IN CORE
       010000
004256 005067           CLR     QSEL             ;QUANT SELECT
       000050
004262 012702'          MOV     #QPARM,R2        ;QUANT PARAMETERS
       004354
004266 012203 LQT2:     MOV     (R2)+,R3         ;ADDRESS INCREMENT
004270 012204           MOV     (R2)+,R4         ;ENTRY COUNT



004272 012205           MOV     (R2)+,R5         ;INITIALIZE ADDRESS
004274 010500 LQT3:     MOV     R5,R0
004276 004767           JSR     PC,DLOAD         ;ADDRESS->D
       000400
004302 004567           JSR     R5,XMI
       000420
004306 000000           +0,716,0                 ;D+0->T
004310 000716
004312 000000
004314 012100           MOV     (R1)+,R0
004316 004767           JSR     PC,DLOAD         ;ENTRY->D
       000360
004322 004567           JSR     R5,XMI2
       000400
004326 000000           +0,12                    ;D->QUANTIZER
004330 000012
004332 000000 QSEL:     +0
```

```
004334 060305           ADD      R3,R5              ;NEXT ADDRESS
004336 005304           DEC      R4                 ;ENTRIES EXHAUSTED?
004340 001355           BNE      LOT3               ;NO
004342 062767           ADD      #10000,OSEL        ;NEXT TABLE
       010000
       177762
004350 100346           BPL      LOT2               ;REPEAT
004352 104420           RETURN

                        ;DA, CNT, A      TABLE GROUP
004354 000002 GFARM:    +BIT1,100,+BIT6 ;  0    3
004356 000100
004360 000100
004362 000002           +BIT1,100,+BIT6 ;  1    3
004364 000100
004366 000100
004370 000004           +BIT2,100,+BIT7 ;  2    [illegible]
004372 000100
004374 000200
004376 000004           +BIT2,100,+BIT7 ;  [illegible]    2
004400 000100
004402 000200
004404 000010           +BIT3,100,[illegible]
004406 000100
004410 000400
004412 000010           +BIT3,100,+BIT8 ;  [illegible]    [illegible]
004414 000100
004416 000400
004420 000010           +BIT3,100,+BIT8 ;  [illegible]    1
004422 000100
004424 000400
004426 000020           +BIT4,400,+BIT11 ;  7    4
004430 000400
004432 004000

       010000 OFARM=10000

004434 012705'JSRPC:    MOV      #NREC,R5
       000134


004440 004735           JSR      PC,@(R5)+
004442 104420           RETURN
```

```
                    ;DUMP REGISTERS AND SCRATCH MEMORY

004444 104422 DUMP:    HTEST
004446 017746          MOV      @MD,-(SP)
       173326
004452 017746          MOV      @MS,-(SP)
       173326
004456 000000'         OCTAL
004460 000030'         +MA,4
004462 000004
004464 000000'         OCTAL
004466 000000'         +MD,4
004470 000004
004472 000000'         OCTAL
004474 000004'         +MS,4
004476 000004
004500 004567          JSR      R5,XMI
       000222
004504 000000          +0,211,0           ;0+0->S
004506 000211
004510 000000
004512 000000'         OCTAL
004514 000004'         +MS,4
004516 000004
004520 000000'         CRLF
004522 005067          CLR      DUMP3
       000004
004526 004567 DUMP2:   JSR      R5,XMI
       000174
004532 000000 DUMP3:   +0,311,0           ;0+R0(B)->S
004534 000311
004536 000000
004540 000000'         OCTAL
004542 000004'         +MS,4
004544 000004
004546 005267          INC      DUMP3
       177760
004552 032767          BIT      #7,DUMP3
       000007
       177752
004560 001001          BNE      .+4
004562 000000'         CRLF
004564 026727          CMP      DUMP3,#20
       177736
       000020
004572 001355          BNE      DUMP2
004574 005067          CLR      DUMP6
       000006
004600 000000'         CRLF
004602 004567 DUMP5:   JSR      R5,XMI
       000120
004606 000000 DUMP6:   +0,1010,0          ;S0->D
004610 001010
```

```
004612 000000
004614 000000'        OCTAL
004616 000000'        +MD,4
004620 000004
004622 062767         ADD     #1000,DUMP6
       001000
       177756
004630 032767         BIT     #7000,DUMP6
       007000
       177750
004636 001001         BNE     .+4
004640 000000'        CRLF
004642 005767         TST     DUMP6
       177740
004646 100355         BPL     DUMP5
004650 012600         MOV     (SP)+,R0
004652 004767         JSR     PC,DLOAD          ;OLD S->D
       000024
004656 004567         JSR     R5,XMI
       000044
004662 000000         +0,711,0          ;D+0->S
004664 000711
004666 000000
004670 012600         MOV     (SP)+,R0
004672 004767         JSR     PC,DLOAD          ;RESTORE D
       000004
004676 000000'        CRLF
004700 104420         RETURN
004702 042700 DLOAD:  BIC     #170000,R0
       170000
004706 010067         MOV     R0,DL2
       000010
004712 004567         JSR     R5,XMI
       000010
004716 000000         +0,7010           ;0->D
004720 007010
004722 000000 DL2:    +0
004724 000207         RTS     PC
              ;EXECUTE MICROINSTRUCTION
004726 010446 XMI:    MOV     R4,-(SP)
004730 016704         MOV     MA,R4
       173074
004734 011466         MOV     (R4),-(SP)     ;SAVE MA
       000030         NOP
004740 005024         CLR     (R4)+          ;MA=0
004742 012446         MOV     (R4)+,-(SP)    ;SAVE MI3
004744 012446         MOV     (R4)+,-(SP)    ;SAVE MI2
004746 011466         MOV     (R4),-(SP)     ;SAVE MI1
       000036         NOP
004752 012514         MOV     (R5)+,(R4)     ;NEW MI1
004754 012544         MOV     (R5)+,-(R4)    ;NEW MI2
004756 012544         MOV     (R5)+,-(R4)    ;NEW MI3
004760 052777         BIS     #BIT5,@MP      ;STEP MODE
       000040
       173034
```

```
004766 005077          CLR     @MGO
       173026
004772 000240          NOP
004774 005077          CLR     @MHALT
       173002
005000 022424          CMP     (R4)+,(R4)+      ;BUMP TO MI1
005002 005077          CLR     @MA
       173022
005006 012614          MOV     (SP)+,(R4)       ;RESTORE MI1
005010 012644          MOV     (SP)+,-(R4)      ;RESTORE MI2
005012 012644          MOV     (SP)+,-(R4)      ;RESTORE MI3
005014 012644          MOV     (SP)+,-(R4)      ;RESTORE MA
005016 012604          MOV     (SP)+,R4
005020 000205          RTS     R5

       000152          .END    BEGIN


ABORT     004152R      AB2       004150R      BD        002350R
BE        002362R      BEGIN     000152R      BELL    = 000007
BIT0    = 000001       BIT1    = 000002       BIT10   = 002000
BIT11   = 004000       BIT12   = 010000       BIT13   = 020000
BIT14   = 040000       BIT15   = 100000       BIT2    = 000004
BIT3    = 000010       BIT4    = 000020       BIT5    = 000040
BIT6    = 000100       BIT7    = 000200       BIT8    = 000400
BIT9    = 001000       BRANCH  = ****** G     BUFFER  = ****** G
CAMCS   = 164100       CAN       003302R      CBEG      000124R
CBN       003536R      CEND      000132R      CERR      003442R
CHAR      000332R      CHAR2     000342R      CIMP      002526R
CIMW      002550R      CIN       003472R      CJMP      002534R
CJMP2     002540R      CJMW      002556R      CJMW2     002562R
CMNNN     003616R      CMP       003672R      CMPL      003720R
CMPL2     003744R      CMPTAB    003762R      CNNNN     003332R
COLUMN  = ****** G     COMMAN    001052R      COMMEN    000366R
COM2      001066R      CPN       003520R      CPOINT    002136R
CR      = 000015       CRLF    = ****** G     CTAB      001204R
CUNN      002272R      CUNNE     002326R      CUNN2     002310R
CXN       003556R      CYN       003570R      DELCH   = 000030
DLINE     000416R      DLOAD     004702R      DL2       004722R
DONOTH    004052R      DUMP      004444R      DUMP2     004526R
DUMP3     004532R      DUMP5     004602R      DUMP6     004606R
ELINE     000412R      ERROR     004044R      EXIT      004120R
FSMCS   = 164100       GO        000424R      GOCH    = 000007
GOX       000560R      GO2       000502R      GO3       000530R
GO4       000562R                             HALT      002464R
HTEST   = 104422       HTEST.    004152RG     HT2       004172R
IM1       000060R      IM2       000100R      JB        002406R
JSRPC     004434R      JZ        002374R      LD        002420R
LE        002420R      LF      = 000012       LINE      000242R
LMPN      004014R      LOAD    = ****** G     LOOK      001102R
LOOKX     001154R      LOOK2     001114R      LOOK3     001134R
```

```
LOOK4    001160R     LOOK5    001176R     LOPT     000056R
LQT      004250R     LQT2     004266R     LQT3     004274R
LSTAT    002432R     LSTEP    000570R     MA       000030RG
MASTER   000656R     MB       000026R     MBASE1 = 164000
MBASE2 = 164040      MBIA     000046R     MCLR     000000R
MCLRCH = 000020      MD       000000R     MEMTRP   004054R
MGO      000020R     MHALT    000002R     MHOLD    002776R
MI       000016R     MIIA     000044R     MI1      000036R
MI2      000034R     MI3      000032R     MJ       000016R
MP       000022R     MS       000004R     MX       000012R
MXIA     000042R     MY       000010R     MYIA     000040R
MZ       000014R     NBEG     000134R     NEND     000146R
NUMBER   000706R     NUMERR   001044R     NUMT     000736R
NUM2     000740R     NUM3     000756R     NUM5     001016R
NUM6     001020R     OCTAL  = ****** G    OSOP     002340R
OSOPD    002332R     OSOPF    002344R     PC      =%000007
PMHOLD   002700R     PNBEG    000150R     PTB      004026R
PTE      004036R     PTFLAG   000120R     PUHOLD   002144R
QIMAGE = 010000      QPARM    004354R     QSEL     004332R
R       =%000000     RCNT     000052R     READ   = ****** G
RETURN = 104420      RETUR.   004074RG    RET2     004112R
RF       002442R     RFLAG    000050R     ROPT     000054R
RS       002442R     RSTAT    002454R     R0      =%000000
R1      =%000001     R2      =%000002     R3      =%000003
R4      =%000004     R5      =%000005     SAVCH    000122R
                     SCAN     000272R     SCAN2    000302R
SNAP     004206R     SNAP2    004240R     SP      =%000006
SPACE    000406R     STEP     000642R     STRING = ****** G
T        002770R     TA       002662R     TAST     003206R
TB       003252R     TD       003216R     TI       003200R
TIM      002572R     TJ       002472R     TJM      002600R
TJM2     002604R     TJM3     002622R     TJ1      002516R
TKS    = 177560      TM       002736R     TMINST   003154R
TMP      003772R     TMP2     004010R     TN       002674R
TNN      003046R     TNNX     003140R     TNN2     003062R
TNN3     003076R     TP       003242R     TS       003230R
TTYBEG = ****** G    TTYEND = ****** G    TU       002156R
TUN      002140R     TUNN     002202R     TUNNX    002270R
TUNN2    002224R     TX       003262R     TY       003272R
TYPE   = ****** G    UHOLD    002162R     UPPER  = ****** G
XMI      004726R     XQTNNN   002626R     XQT2     002652R
.      = 005022R
```

MODIFIED EXECUTE MICROINSTRUCTION

```
7400  010446     XMIZ:  MOV   R4,-(SP)
7402  013704            MOV   @#MA,R4
      001030
7406  011446            MOV   (R4),-(SP)    ;SAVE MA
7410  005024            CLR   (R4)+         ;MA=0
7412  012446            MOV   (R4)+,-(SP)   ;SAVE MI AT LOC 0
7414  012446            MOV   (R4)+,-(SP)
7416  011446            MOV   (R4),-(SP)
7420  005014            CLR   (R4)          ;NOP  LOC 0
7422  012744            MOV   #10,-(R4)
      000010
7426  005044            CLR   -(R4)
7430  012744            MOV   #1,-(R4)      ;MA=1
      000001
7434  005724            TST   (R4)+
7436  012446            MOV   (R4)+,-(SP)   ;SAVE MI AT LOC 1
7440  012446            MOV   (R4)+,-(SP)
7442  011446            MOV   (R4),-(SP)
7444  012514            MOV   (R5)+,(R4)    ;NEW MI  LOC 1
7446  012544            MOV   (R5)+,-(R4)
7450  012544            MOV   (R5)+,-(R4)
7452  005044            CLR   -(R4)         ;MA=0
7454  017746            MOV   @MP,-(SP)
      171342-
7460  012777            MOV   #40,@MP
      000040
      171334-
7466  005077            CLR   @M60
      171326-
7472  000240            NOP
7474  005077            CLR   @M60
      171320-
7500  000240            NOP
7502  012677            MOV   (SP)+,@MP
      171314-
```

```
7506  012724            MOV  #1,(R4)+
      000001
7512  022424            CMP  (R4)+,(R4)+
7514  012614            MOV  (SP)+,(R4)
7516  012614            MOV  (SP)+,-(R4)
7520  012644            MOV  (SP)+,-(R4)
7522  005044            CLR  -(R4)
7524  062704            ADD  #6,R4
      000006
7530  012614            MOV  (SP)+,(R4)
7532  012644            MOV  (SP)+,-(R4)
7534  012644            MOV  (SP)+,-(R4)
7536  012644            MOV  (SP)+,-(R4)
7540  012604            MOV  (SP)+,R4
7542  000205            RTS  R5
```

```
                .TITLE LOAD   MICROPROGRAM

                ; JSR PC,LOAD
                ;RELOCATION BASE IS IN R5

                .GLOBL LOAD,IN,MA,OCTAL,CRLF,STRING
         000015 CR=15
         000007 BELL=7
         000000 R=%0
         000001 R1=%1
         000002 R2=%2
         000003 R3=%3
         000004 A=%4
         000005 BASE=%5
         000006 SP=%6
         000007 PC=%7

                BUF:
         000014 .=.+12.

000014 004567 LOAD:     JSR     %5,INP          ;***
       000344
000020 022700           CMP     #1,R    ;
       000001
000024 001373           BNE     LOAD    ;SCAN AWAY LEADER
000026 016704'          MOV     MA,A
       000000
000032 011446           MOV     (A),-(SP) ;SAVE MA
000034 010514           MOV     BASE,(A)  ;START AT RELOCATION BASE
000036 010527           MOV     BASE,(PC)+
000040 000000 BLOCK:    +0                ;BEGINNING OF BLOCK
000042 005027           CLR     (PC)+
000044 000000 CHKSUM:   +0

000046 012701'LOOPA:    MOV     #BUF,R1
       000000

000052 004567 LOOPB:    JSR     %5,INP
       000306
000056 060067           ADD     R,CHKSUM
       177762
000062 110021           MOVB    R,(R1)+   ;READ 6 CHARACTERS
000064 020127'          CMP     R1,#BUF+6 ;INTO BUF TO BUF+5
       000006
000070 100770           BMI     LOOPB


000072 026727           CMP     BUF+2,#160000
       177704
       160000
000100 103026           BHIS    CTRL      ;CONTROL WORD

000102 012702'          MOV     #BUF+12.,R2
       000014
```

```
000106 016703'          MOV     MA+2,R3
       000002
000112 014113           MOV     -(R1),(R3) ;MOVE TO CONTROL
000114 012342           MOV     (R3)+,-(R2);STORE AND READ
000116 014113           MOV     -(R1),(R3) ;BACK INTO BUF+6
000120 012342           MOV     (R3)+,-(R2);TO BUF+11.
000122 014113           MOV     -(R1),(R3)
000124 012342           MOV     (R3)+,-(R2)

000126 022122           CMP     (R1)+,(R2)+ ;CHECK FOR ERROR
000130 001073           BNE     ERR         ;IN READBACK
000132 022122           CMP     (R1)+,(R2)+
000134 001071           BNE     ERR
000136 022122           CMP     (R1)+,(R2)+
000140 001067           BNE     ERR

000142 011446 OUT:      MOV     (A),-(SP)
000144 042716           BIC     #176000,(SP)
       176000
000150 005216           INC     (SP)
000152 012614           MOV     (SP)+,(A)        ;NEXT UINSTR TO NEXT LOC
000154 000734           BR      LOOPA

000156 021437 CTRL:     CMP     (A),@(PC)+ ;COMPARE CURRENT ADDRESS
000160 000040'PBLOCK:   +BLOCK             ;WITH BEGINNING OF BLOCK
000162 001410           BEQ     CTRL2      ;NO UINSTR IN BLOCK
000164 005314           DEC     (A)
000166 000000'          OCTAL
000170 000160'          +PBLOCK,4
000172 000004
000174 000000'          OCTAL
000176 000000'          +MA,4
000200 000004
000202 000000'          CRLF        ;TYPE OUT BLOCK LIMITS

000204 016700 CTRL2:    MOV     BUF+4,R
       177574
000210 060500           ADD     BASE,R
000212 042700           BIC     #176000,R
       176000
000216 010014           MOV     R,(A)      ;CHANGE MA
000220 010067           MOV     R,BLOCK    ;RESET BLOCK BEGINNING
       177614
000224 032767           BIT     #10000,BUF+2
       010000
       177550
000232 001705           BEQ     LOOPA      ;CONTINUE READING TAPE

000234 004567           JSR     %5,INP           ;GET CHKSUM
       000124
000240 120067           CMPB    R,CHKSUM   ;CHECK IT
       177600
000244 001412           BEQ     CTRL3      ;IT CHECKS

000246 000000'          STRING
000250    103           .ASCII  /CHECKSUM ERROR/
```

```
000251    110
000252    105
000253    103
000254    113
000255    123
000256    125
000257    115
000260    040
000261    105
000262    122
000263    122
000264    117
000265    122
000266    007               .BYTE   BELL,BELL,CR,0
000267    007
000270    015
000271    000
       000272               .EVEN

000272 000000'CTRL3:        STRING
000274    114               .ASCII  /LOAD COMPLETED/
000275    117
000276    101
000277    104
000300    040
000301    103
000302    117
000303    115
000304    120
000305    114
000306    105
000307    124
000310    105
000311    104
000312    015               .BYTE   CR,0
000313    000
       000314               .EVEN

000314 012614 EXIT:         MOV     (SP)+,(A)   ;RESTORE MA
000316 000207               RTS     PC

000320 000000'ERR:          OCTAL               ;TYPE OUT BAD LOCATION
000322 000000'              +MA,4
000324 000004
000326 012727'              MOV     #BUF,(PC)+
       000000
000332 000000'POINT:        +BUF
000334 000000'ERR2:         OCTAL
000336 000332'              +POINT,6            ;TYPE OUT INSTR LOADED
000340 000006
000342 062767               ADD     #2,POINT    ;AND INS[illegible]
       000002
       177767
[illegible] 026727'         CMP     POINT,#BUF+14.
       177756
       000014
```

```
000356 100776             BMI     ERR2
000360 000000'            CLCF
000362 000667             BR      OUT
                 .GLOBL   IN2
000364 016746'IN2:        MOV     DRIN,-(SP)
       167734
000370 012767'            MOV     #1,DROT
       000001
       167732
000376 012700             MOV     #1,%0
       000001
000402 004567             JSR     %5,DELAY
       000026
000406 012767'            MOV     #0,DROT
       000000
       167732
000414 012700             MOV     #15,%0
       000015
000420 004567             JSR     %5,DELAY
       000010
000424 012600             MOV     (SP)+,%0
000426 042700             BIC     #177400,%0
       177400
000432 000205             RTS     %5
000434 012767 DELAY:      MOV     #DLY,CNT
       000100
       000014
000442 005367 DX:         DEC     CNT
       000010
000446 001375             BNE     DX
000450 005300             DEC     %0
000452 001370             BNE     DELAY
000454 000205             RTS     %5
       000100 DLY=100

       167734 DRIN=167734
       167732 DROT=167732
000456 000000 CNT:+0

       000001             .END
```

```
                .TITLE TTY    TELETYPE I/O

         000000 R0=%0
         000001 R1=%1
         000005 R5=%5
         000006 SP=%6
         000007 PC=%7
         000015 CR=15
         000012 LF=12
         000007 BELL=7
         000177 RUB=177
         177560 TKS=177560
         177562 TKB=177562
         177564 TPS=177564
         177566 TPB=177566

                .GLOBL IN,OUT,READ,TYPE
         104400 T=104400
         104400 IN=T
         104402 OUT=T+2
         104404 READ=T+4
         104406 TYPE=T+6


                ; TRAP 0  READ CHARACTER INTO R0 W/O EDITING

                            .GLOBL  IN.
000000 105737 IN.:          TSTB    @#TKS
       177560
000004 100375               BPL     IN.               ;WAIT FOR CHARACTER
000006 113700               MOVB    @#TKB,R0          ;GET CHARACTER
       177562
000012 005237               INC     @#TKS             ;RESET FLAG
       177560
000016 042700               BIC     #177400,R0        ;CLEAR TOP BYTE
       177400
000022 000205               RTS     R5

                ; TRAP 2  TYPES OR PUNCHES CHARACTER IN R0 W/O EDITING

                            .GLOBL  OUT.

000024 105737 OUT.:         TSTB    @#TPS
       177564
000030 100375               BPL     OUT.              ;WAIT UNTIL TTY IS READY
000032 110037               MOVB    R0,@#TPB          ;SEND CHARACTER
       177566
000036 000205               RTS     R5

                ; TRAP 4  READ CHARACTER INTO R0 AND HANDLE
                ;         SPECIAL CHARACTERS

                            .GLOBL  READ.,BRANCH,COLUMN,BUFFER
```

```
000040 104400 READ.:  IN                          ;GET CHARACTER
000042 042700         BIC     #200,R0             ;CONVERT TO ASCII
       000200
000046 000000'        BRANCH
000050    000         .BYTE   0,IGNORE-.
000051    031
000052    012         .BYTE   LF,IGNORE-.
000053    027
000054    177         .BYTE   RUB,IGNORE-.
000055    025
000056    004         .BYTE   'D-100,DITTO-. ;CTRL D
000057    007
000060    005         .BYTE   'E-100,ECHO-.  ;CTPL E
000061    015
000062    006         .BYTE   'F-100,NECHO-. ;CTRL F
000063    011
000064    000         .BYTE   0,EXIT-.
000065    005

000066 117700 DITTO:  MOVB    @COLUMN,R0 ;COPY FROM LINE ABOVE
       000042
000072 000205 EXIT:   RTS     R5
000074 005000 NECHO:  CLR     R0
000076 010027 ECHO:   MOV     R0,#1
       000001
       000100 TFLAG=.-2
000102 000756 IGNORE: BR      READ.

              ; TRAP 6  TYPE CHARACTER AND @NDLE
              ;         SPECIAL CHARS, UPDATE LINE BUFFER

                      .GLOBL  TYPE.

000104 005767 TYPE.:  TST     TFLAG
       177770
000110 001003         BNE     DOTYPE              ;FLAG SET
000112 020027         CMP     R0,#BELL            ;RING BELL EVEN IF TFLAG=0
       000007
000116 001015         BNE     TEXIT               ;DO NOT TYPE IF TFLAG=0
000120 000000'DOTYPE: BRANCH
000122    007         .BYTE   BELL,TBELL-.        ;DO NOT SAVE BELL
000123    031
000124    015         .BYTE   CR,TCR-.            ;CR IS SPECIAL
000125    033
000126    000         .BYTE   0,1
000127    001
000130 022727'        CMP     #BUFFER+72.,#BUFFER ;CHECK FOR RUNNING OFF END
       000346
       000236
       000134 COLUMN=.-2
000136 001427         BEQ     OFFEND              ;RUNNING OFF END
000140 104402 TOUT:   OUT                         ;TYPE CHARACTER
000142 110077         MOVB    R0,@COLUMN
       177766
000146 005267         INC     COLUMN
       177762
```

```
000152 000205 TEXIT:  RTS     R5
000154 104402 TBELL:  OUT                     ;RING BELL
000156 000205         RTS     R5
000160 004767 TCR:    JSR     PC,TCRLF
       000006
000164 012700         MOV     #CR,R0
       000015
000170 000205         RTS     R5
000172 012700 TCRLF:  MOV     #CR,R0          ;TYPE CR LF
       000015
000176 104402         OUT
000200 012700         MOV     #LF,R0
       000012
000204 104402         OUT
000206 012767'        MOV     #BUFFER,COLUMN  ;RESET COLUMN POINTER
       000236  6740
       177720
000214 000207         RTS     PC
000216 010046 OFFEND: MOV     R0,-(SP)        ;SAVE R0
000220 012700         MOV     #BELL,R0
       000007
000224 104402         OUT                     ;RING BELL
000226 004767         JSR     PC,TCRLF        ;GO TO NEXT LINE
       177740
000232 012600         MOV     (SP)+,R0        ;RESTORE R0
000234 000741         BR      TOUT            ;TYPE CHAR ON NEXT LINE
000236    040 BUFFER: .ASCII /
000237    040
000240    040
000241    040
000242    040
000243    040
000244    040
000245    040
000246    040
000247    040
000250    040
000251    040
000252    040
000253    040
000254    040
000255    040
000256    040
000257    040
000260    040
000261    040
000262    040         .ASCII /
000263    040
000264    040
000265    040
000266    040
000267    040
000270    040
000271    040
000272    040
000273    040
```

where B_i and C_i are discrete Fourier transforms of length 32.

```
000274   040
000275   040
000276   040
000277   040
000300   040
000301   040
000302   040
000303   040
000304   040
000305   040
000306   040             .ASCII /
000307   040
000310   040
000311   040
000312   040
000313   040
000314   040
000315   040
000316   040
000317   040
000320   040
000321   040
000322   040
000323   040
000324   040
000325   040
000326   040
000327   040
000330   040
000331   040
000332   040             .ASCII /                  / ; 74 SPACES
000333   040
000334   040
000335   040
000336   040
000337   040
000340   040
000341   040
000342   040
000343   040
000344   040
000345   040
000346   040
000347   040

      000001             .END
```

```
                .TITLE EDOUT  EDITED OUTPUT

        000000 R0=%0
        000001 R1=%1
        000002 R2=%2
        000005 R5=%5
        000006 SP=%6

                .GLOBL CRLF,OCTAL,STRING,CRLF.,OCTAL.,STRIN.,TYPE
        104400 T=104400
        104412 CRLF=T+12
        104414 OCTAL=T+14
        104416 STRING=T+16

                ; TRAP 12  TYPES CR LF

000000 104416 CRLF.:  STRING
000002    015         .BYTE   15,0
000003    000
000004 000205         RTS     R5

                ; TRAP 14
                ; +POINTER,N
                ; TYPES DATA IN N OCTAL DIGITS, PRECEDED BY A
                ; SPACE.  POINTER IS ADDRESS OF ADDRESS
                ; OF DATA.

000006 010046 OCTAL.: MOV     R0,-(SP)
000010 010146         MOV     R1,-(SP)
000012 010246         MOV     R2,-(SP)  ;SAVE REGISTERS
000014 013500         MOV     @(R5)+,R0
000016 011000         MOV     (R0),R0   ;DATA->R0
000020 012501         MOV     (R5)+,R1  ;N->R1
000022 062701'        ADD     #BUF+2,R1 ;R1 POINTS TO END
       000070
000026 105041         CLRB    -(R1)     ;ZERO END OF BUFFER
000030 010002 LOOPA:  MOV     R0,R2
000032 042702         BIC     #177770,R2 ;GET 3 LSB'S
       177770
000036 062702         ADD     #'0,R2    ;CONVERT TO ASCII
       000060
000042 110241         MOVB    R2,-(R1)  ;PUT INTO BUFFER
000044 006200         ASR     R0
000046 042700         BIC     #100000,R0
       100000
000052 006200         ASR     R0
000054 006200         ASR     R0        ;SHIFT RIGHT 3 BITS
000056 022701'        CMP     #BUF+1,R1 ;CHECK FOR END
       000067
000062 100762         BMI     LOOPA
000064 104416         STRING
000066    040 BUF:    .ASCII  / /
000067    040
000070 000000         +0
000072 000240         NOP
```

```
000074 000240          NOP
000076 012767          MOV     #240,BUF+4
       000240
       177766
000104 012767          MOV     #240,BUF+6 ;PATCH UP NOP'S
       000240
       177762
000112 012602          MOV     (SP)+,R2
000114 012601          MOV     (SP)+,R1
000116 012600          MOV     (SP)+,R0   ;RESTORE REGISTERS
000120 000205          RTS     R5

               ; TRAP 16
               ; .ASCII /MESSAGE/
               ; .BYTE 0
               ; .EVEN
               ; TYPES MESSAGE

000122 010046 STRIN.: MOV     R0,-(SP)
000124 005000          CLR     R0
000126 112500 LOOPB:   MOVB    (R5)+,R0   ;GET CHARACTER
000130 001402          BEQ     OUTB       ;STOP IF CHARACTER IS NULL
000132 000000'         TYPE
000134 000774          BR      LOOPB
000136 010500 OUTB:    MOV     R5,R0
000140 006000          ROR     R0
000142 005505          ADC     R5         ;INCREMENT R5 IF IT IS ODD
000144 012600          MOV     (SP)+,R0
000146 000205          RTS     R5

       000001          .END
```

```
                    .TITLE BRANCH  ON CHARACTER

                    ; JSR    R5,BRANCH
                    ; .BYTE C1,A1-.
                    ; .BYTE C2,A2-.
                    ;     * * *
                    ; .BYTE CN,AN-.
                    ; .BYTE 0,A-.
                    ;
                    ; IF (R0)=CK,BRANCH TO AK
                    ; OTHERWISE BRANCH TO A
                    ; NOTE C1=0 IS PERMITTED

             000000 R0=%0
             000001 R1=%1
             000005 R5=%5
             000006 SP=%6

                             .GLOBL  BRANCH,BRANC.
             104410 BRANCH=104410


      000000 010146 BRANC.: MOV      R1,-(SP);SAVE R1
      000002 005001         CLR      R1
      000004 112501         MOVB     (R5)+,R1;C1->R1
      000006 000402         BR       LOOPB
      000010 112501 LOOP:   MOVB     (R5)+,R1;CK->R1
      000012 001404         BEQ      EXIT    ;EXIT IF CK=0, K NE 1
      000014 020001 LOOPB:  CMP      R0,R1
      000016 001402         BEQ      EXIT    ;EXIT IF R0=CK
      000020 005205         INC      R5      ;OTHERWISE SKIP AK-.
      000022 000772         BR       LOOP
```

```
000024  000137                 EXIT:  JMP   @#7330
000026  007330

007330  111501                        MOVB  (R5),R1 ;AK-.
007332  042701                        BIC   #177400,R1
        177400
007336  060105                        ADD   R1,R5 ;ADD.
007340  012601                        MOV   (SP)+,R1
007342  000205                        RTS   R5
```

```
                .TITLE TRAP   PROCESSOR

                ; MAKES TRAP LOOK LIKE JSR R5, EXCEPT
                ; THAT PS IS DESTROYED

        000005 R5=%5
        000006 SP=%6
        000007 PC=%7

        000000          .ASECT
        000034 .=34
000034  000000'         +TPROC
000036  000000          +0        ;NEW PS


        000000          .CSECT
000000  010566 TPROC:   MOV       R5,2(SP)         ;WRITE R5 OVER PS
        000002
000004  011605          MOV       (SP),R5          ;RETURN POINT->R5
000006  016505          MOV       -2(R5),R5        ;GET TRAP INSTRUCTION
        177776
000012  016567'         MOV       TABLE-104400(R5),JUMP+2
        073426
        000004
000020  012605          MOV       (SP)+,R5         ;RETURN POINT->R5
000022  012707 JUMP:    MOV       #0,PC            ;JUMP
        000000

               TABLE:
                        .GLOBL    IN.,OUT.,READ.,TYPE.
000026  000000'         .WORD     IN.,OUT.,READ.,TYPE.
000030  000000'
000032  000000'
000034  000000'
                        .GLOBL    BRANC.
000036  000000'         .WORD     BRANC.
                        .GLOBL    CRLF.,OCTAL.,STRIN.
000040  000000'         .WORD     CRLF.,OCTAL.,STRIN.
000042  000000'
000044  000000'
                        .GLOBL    RETUR.,HTEST.
000046  000000'         .WORD     RETUR.,HTEST.
000050  000000'
                        .GLOBL    UPPER
000052  000000 UPPER:   +0        ;UPPER LIMIT OF OS CODE

        000001          .END
```

```
TRANSFER ADDRESS: 001152
LOW LIMIT: 001000
HIGH LIMIT: 007332
**********
MODULE  MAIN
SECTION ENTRY   ADDRESS SIZE
<. ABS.>        000000  000000
<      >.       001000  005022
        HTEST.  005152
        MA      001030
        RETUR.  005074
**********
MODULE  LOAD
SECTION ENTRY   ADDRESS SIZE
<      >        006022  000460
        INP     006406
        LOAD    006036
**********
MODULE  TTY
SECTION ENTRY   ADDRESS SIZE
<      >        006502  000350
        BUFFER  006740
        COLUMN  006636
        IN      104400
        IN.     006502
        OUT     104402
        OUT.    006526
        READ    104404
        READ.   006542
        TYPE    104406
        TYPE.   006606
**********
MODULE  EDOUT
SECTION ENTRY   ADDRESS SIZE
<      >        007052  000150
        CRLF    104412
        CRLF.   007052
        OCTAL   104414
        OCTAL.  007060
        STRING  104416
        STRIN.  007174
**********
MODULE  BRANCH
SECTION ENTRY   ADDRESS SIZE
<      >        007222  000034
        BRANCH  104410
        BRANC.  007222
**********
MODULE  TRAP
SECTION ENTRY   ADDRESS SIZE
<      >        007256  000054
        UPPER   007330
```

ATTACHMENT 3 TO APPENDIX B

PAPER TAPE INDEX

Label	Loaded by
ABSOLUTE LOADER/H.S. BOOT	Bootstrap
OPERATING SYSTEM	Absolute Loader
DCT-DPCM (FORWARE & INVERXE)	LMP commands
TRANSFORM PATCHES FOR 1.6 BITS/PIXEL	LMP commands
TRANSFORM PATCHES FOR .8 BITS/PIXEL	LMP commands
TRANSFORM PATCHES FOR .4 BITS/PIXEL	LMP commands
CONTROL STORE PUNCH	Absolute Loader
COMPLEMENT/COMPLEMENT	LMP commands
FIRST PIXEL FIX	LMP commands
FIRST PIXEL UNFIX	LMP commands
ABSOLUTE PUNCH	Absolute Loader
DEMI/DEMO	Absolute Loader
DEMONSTRATION DRIVER	Absolute Loader
DEMONSTRATION TAPE (1240 OBJECT TAPE)	LMP commands

NOTE: the CONTROL STORE PUNCH is included on the DWD 1240 SYSTEM TAPE and has to be loaded separately only to restore it after user routines have overstored it.

APPENDIX C

TV REDUNDANCY REDUCTION SYSTEM USING A BIPOLAR MICROPROCESSOR

DD1423 Item #0001

Contract #N66001-76-C-0080

T-36-928

Prepared for
Naval Undersea Center
San Diego, CA 92132
March 2, 1976

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

Theoretical and experimental work at both the University of Southern California and the Naval Undersea Center have shown that a hybrid compression scheme consisting of a Discrete Cosine Transform (DCT) along a TV line and Differential Pulse Code Modulation (DPCM) line to line is a near optimum way of achieving redundancy reduction in conventional TV. The compression is partly achieved by lowering the field rate by a factor of 8 so that a stripe which is only 32 pixels, rather than 256 pixels wide, is processed out of each field. The DCT/DPCM provides the rest of the data compression and can lower the bit rate to 200,000 bits/sec. A complete system including both compression and the corresponding expansion system is depicted in Fig. C-1. It is the top half of this diagram with which the present report is concerned.

The mechanization to be described consists of three 4-bit Am 2901 bipolar microprocessor "slices" together with suitable I/O, RAM, ROM, and high-speed multiplier. Without simulation it is impossible to determine whether 8 bits of precision would have been adequate; hence 12 bits were used. The pre-emphasis filter which is shown is known to be required in an analog implementation using charge-coupled devices because of dynamic range problems. In the system to be described, this filter is not needed. In its place there is a high performance A/D converter.

It is of utmost importance to have an efficient algorithm for computation of the DCT. Data/Ware Development has developed such an algorithm which is employed in this implementation. The resulting system is small in size, yet powerful.

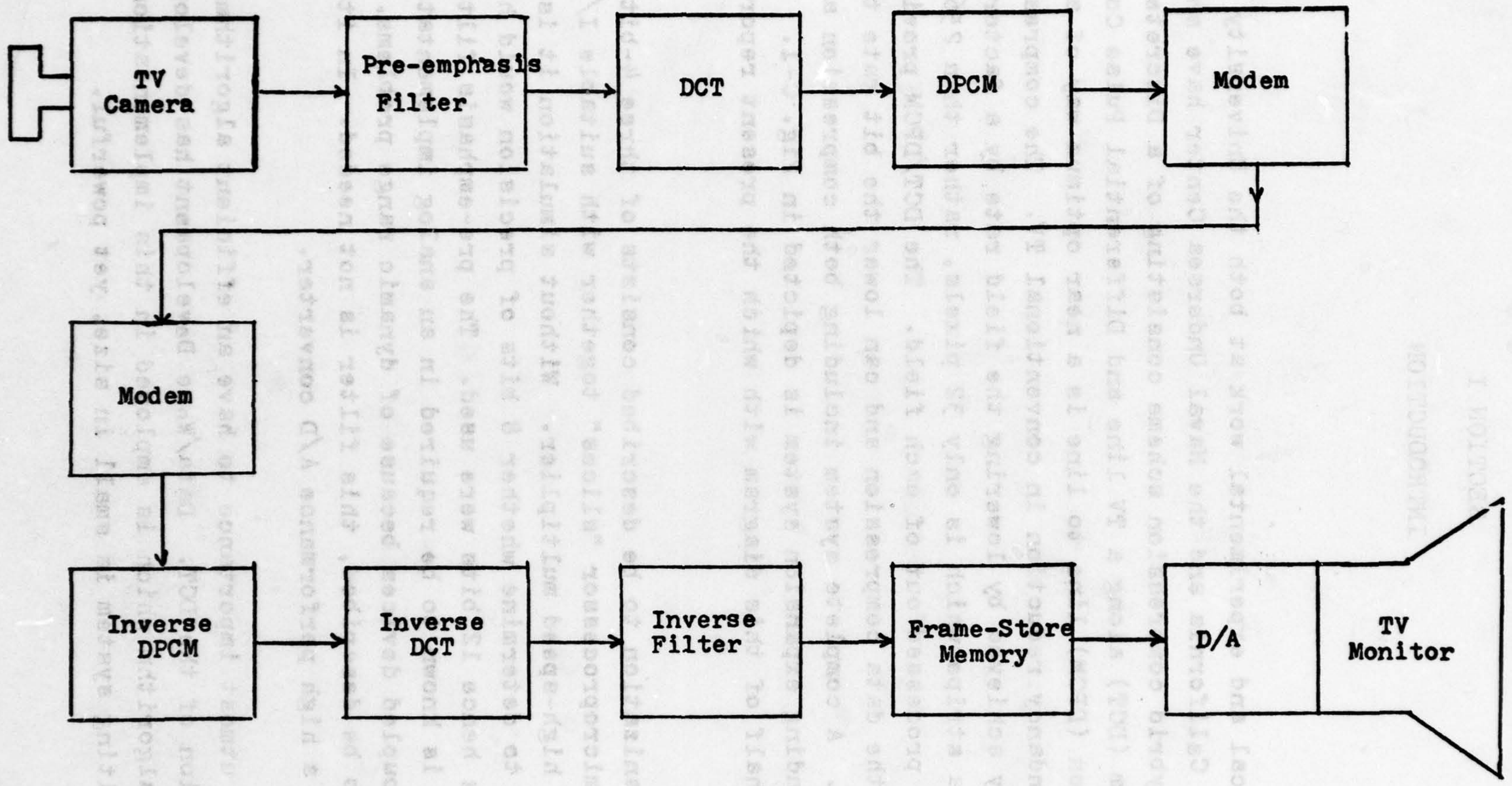


Figure C-1. TV Imagery Data Compression System

SECTION II
DCT AND DPCM ALGORITHM

2.1 DCT Mathematics

If $g_0, g_1, \ldots, g_{31}$ are the numbers representing the pixel values in one "strip" of a TV line, the Discrete Cosine Transform (DCT) is given by

$$G_k = 2\sum_{j=0}^{31} g_j \cos\left[(j+\tfrac{1}{2})k\theta\right]$$

$$k = 0, 1, \ldots, 31$$

where $\theta = 2\pi/64$. By combining complex conjugate terms,

$$G_k = w^{-\frac{1}{2}k} \sum_{j=0}^{63} a_j w^{-jk} \qquad \text{(C-1)}$$

where $w = e^{i\theta}$ and

$$a_j = \begin{cases} g_j & \text{if } j \leq 31 \\ g_{63-j} & \text{if } j \geq 32 \end{cases}$$

This formulation can be simplified further.

Define

$$A_k = \sum_{j=0}^{63} a_j w^{-jk}$$

Then $G_k = w^{-\frac{1}{2}k} A_k$ and the A_k are a discrete Fourier transform of length 64 with real inputs. Appendix A describes a method of calculating such transforms due to Data/Ware Development which is better than other methods in the literature. In the last step of that method, the A_k are computed as follows:

$$A_k = B_k + w^{-k} C_k, \quad k = 0, 1, \ldots, 16$$

$$\bar{A}_{32-k} = B_k - w^{-k} C_k, \quad k = 0, 1, \ldots, 16$$

where B_k and C_k are discrete Fourier transforms of length 32. It can be shown that

$$G_k = w^{-\frac{1}{2}k} A_k = w^{-\frac{1}{2}k} B_k + w^{-\frac{3}{2}k} C_k$$

$$-iG_{32-k} = w^{-\frac{1}{2}k} \bar{A}_{32-k} = w^{-\frac{1}{2}k} B_k - w^{-\frac{3}{2}k} C_k$$

The sum of these two equations is

$$G_k - iG_{32-k} = 2w^{-\frac{1}{2}k} B_k \tag{C-2}$$

and hence

$$G_k = 2\mathrm{Re}(w^{-\frac{1}{2}k} B_k)$$

$$k = 0, 1, \ldots, 16 \tag{C-3}$$

$$G_{32-k} = -2\mathrm{Im}(w^{-\frac{1}{2}k} B_k)$$

$$k = 1, 2, \ldots, 15 \tag{C-4}$$

which expresses the desired transform in terms of the B_k, which are the discrete Fourier transform of g_0, g_2, g_4, ..., g_{30}, g_{31}, g_{29}, g_{27}, ..., g_1.

The B_k can be computed by the method described in Attachment 1. The computations are illustrated in Figure C-1.

The operation count for computing the B_k is shown in Table C-1. The only nontrivial values of w^{-k} used in the computations are those corresponding to k = 2, 4, 6, 8, 10, 12, 14. The components of these values may be taken from a table of $\sin 2k\theta$, since $\cos 2k\theta = \sin((16-2k)\theta)$. Hence there are only seven distinct nontrivial multipliers. The factor of 2 in (C-3) and (C-4) is absorbed into the output scaling.

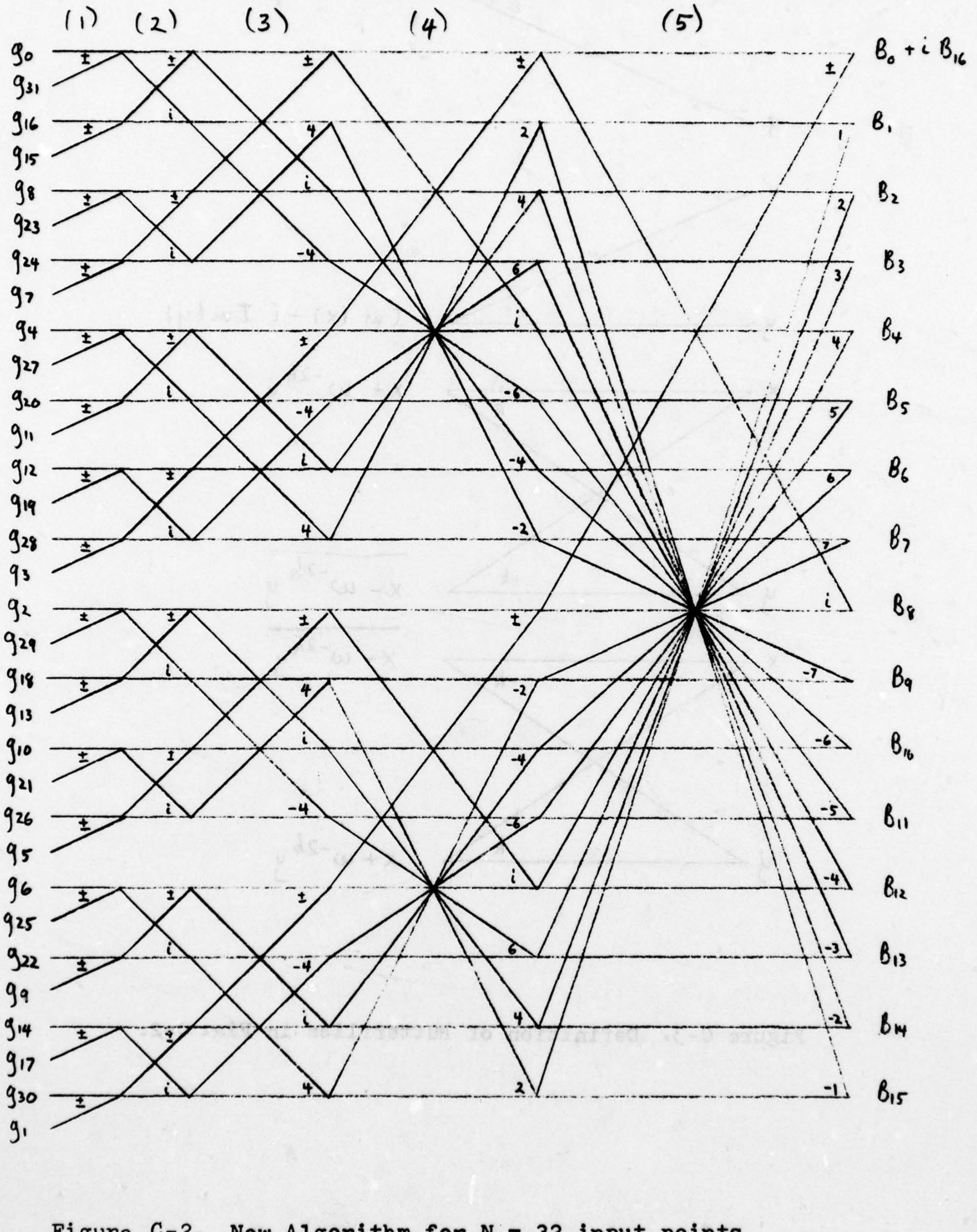


Figure C-2. New Algorithm for N = 32 input points

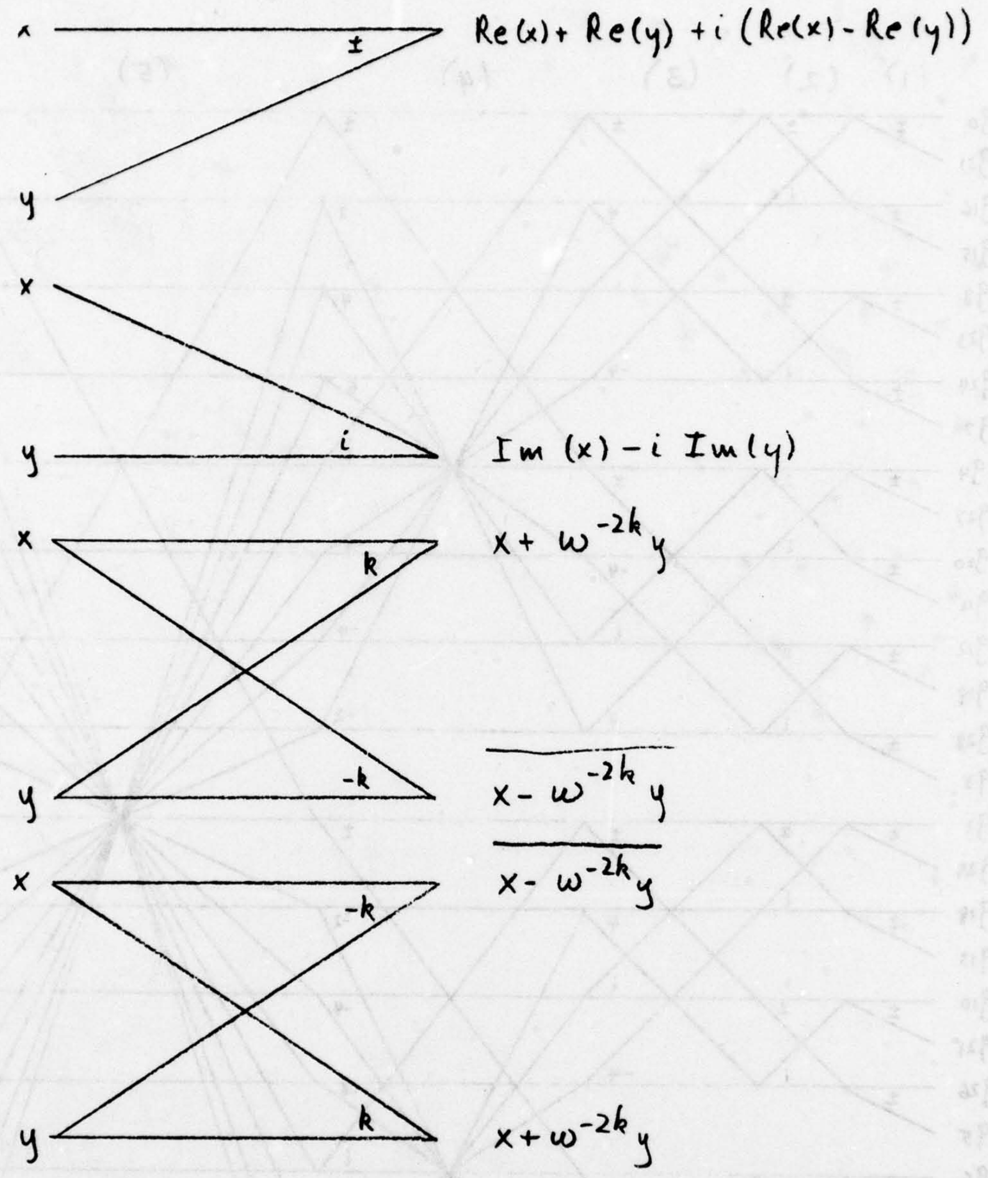

x
y
±
Re(x)+ Re(y) +i (Re(x)- Re(y))
x
y
i
Im (x) - i Im(y)
x
y
k
-k
$x + \omega^{-2k} y$
$\overline{x - \omega^{-2k} y}$
x
y
-k
k
$\overline{x - \omega^{-2k} y}$
$x + \omega^{-2k} y$

Figure C-3. Definition of Butterflies in Fig. C-2.

TABLE C-1

OPERATION COUNTS FOR

COMPUTING THE B_k AS ILLUSTRATED IN FIGURE C-2

Symbol(s)	Number of Occurrences	Operations Per Occurrence		Total Number of Operations	
		add	multiply	add	multiply
$\pm$	31	2	0	62	0
i	15	0	0	0	0
4,-4 pair	7	6	2	42	14
k,-k pair, k≠4	10	6	4	60	40
Totals				164	54

Because of the close relationship between the B_k and the G_k shown by equations (C-3) and (C-4), it may be sufficient to transmit the components of the B_k, instead of the G_k. The question is discussed more fully in the next section.

2.2 Final Rotations

The final rotations in (C-3) and (C-4) may be omitted with perhaps only a slight degradation in the quality of the transform for image compression. The heuristic arguments given here are not sufficient to decide the question, and some experimentation or simulation will be required.

The advantage of the DCT lies in the facts that the DCT coefficients have Gaussian distribution, under a certain model, and that their variances and covariances are close to those of the "optimal" Karhunen-Loeve transform (KLT) coefficients, which are independent and otherwise "optimally" distributed. Therefore, the G_k have variances close to those of the KLT coefficients and they are nearly independent.

From (C-2)

$$B_k = \tfrac{1}{2}w^{\frac{1}{2}k}(G_k - iG_{32-k})$$

or

$$\text{Re } B_k = \tfrac{1}{2}\, G_k \cos(\tfrac{1}{2}k\theta) + \tfrac{1}{2}\, G_{32-k} \sin(\tfrac{1}{2}k\theta)$$

$$\text{Im } B_k = \tfrac{1}{2}\, G_k \sin(\tfrac{1}{2}k\theta) - \tfrac{1}{2}\, G_{32-k} \cos(\tfrac{1}{2}k\theta)$$

where $\theta = \pi/32$, $k = 0, 1, \ldots, 16$

The angle $\frac{1}{2}k\theta$ is never more than 45°. Moreover, when θ is large, the variances of G_k and G_{32-k} are nearly equal for most pictures; and it is easily shown that, except for the factor $\frac{1}{2}$, the variances of Re B_k and Im B_k are nearly the same and they are nearly independent. (They would have exactly equal variances and and exactly zero covariance if G_k and G_{32-k} had exactly equal variances and exactly zero covariance.) Hence Re B_k and Im B_k are "good" substitutes for G_k and G_{32-k}.

When θ is small, Re $B_k \cong \frac{1}{2}G_k$ and Im $B_k \cong -G_{32-k}$, so once again the substitution of Re B_k and Im B_k for G_k and G_{32-k}, respectively is "good" in the sense considered here.

If the G_k are to be transmitted, then the products $w^{-\frac{1}{2}k}$ B_k for $k = 0, 1, \ldots, 16$ must be formed. The operation for $k = 0$ is trivial, and for $k = 16$ only the real part of the product is necessary. Multiplication by the components of all such $w^{-\frac{1}{2}k}$ is required, and there are 31 such components (note: $\cos\frac{1}{2}k\theta = \sin\frac{1}{2}k\theta$ when $k = 16$), which include all those needed to compute the B_k.

To compute $w^{-\frac{1}{2}k}$ B_k requires four multiplications and two additions for each $k = 1, 2, \ldots, 15$. To compute Re $(w^{-\frac{1}{2}k} B_k)$ when $k = 16$ requires only one multiplication, since B_{16} is real. Hence 61 multiplications and 30 additions are required to complete the computations.

2.3 DPCM

Each DCT coefficient (or the substitute described in Section 2.2) is quantized and transmitted according to a Differential Pulse Code Modulation (DPCM) scheme. The particular quantization values used vary according to the coefficient and the output bit rate. A complete list of quantization schemes is available in Appendix B, but their assignment to the various DCT coefficients and output bit rates is not available at this time.

The quantizer accepts an input value x and produces two output values, according to the tables given in Attachment 2. The rounded value R(x) is given in the same kind of arithmetic used for other computations. The output code O(x) varies from zero to six bits according to the same tables. (A zero-bit output code is vacuous, and no information is sent to the modem in this case, but it is easier to compute the corresponding DCT coefficient and DPCM output than to program the system to avoid such computations.) The quantization schemes used are those that are believed to be "optimal" for most kinds of pictures.

Let $G_k^{(n)}$ be a particular DCT coefficient associated with the n-th 32-pixel burst. Then the DPCM is defined by Figure C-3, or in symbols as follows:

$$x = G_k^{(n)} - x\,\hat{G}_k^{(n-1)} \qquad \text{(C-5)}$$

$$\text{output} = O\,(x) \qquad \text{(C-6)}$$

$$\hat{G}_k^{(n)} = R(x) + x\,\hat{G}_k^{(n-1)} \qquad \text{(C-7)}$$

The "starting" value of $\hat{G}_k^{(0)}$ is immaterial, since $0 < \alpha < 1$ and errors are attenuated.

All inputs g_j to the DCT are 6-bit twos complement integers. Hence $-32 \leq g_j \leq 31$, and it then is obvious from the definition of the DCT coefficients G_k that $|G_k| \leq 2.32.\max|g_j| = 2{,}048$. Since the factor of 2 in (C-3) and (C-4) is not used, the transmitted values of G_k have absolute value no greater than 1,024, which comfortably fits into 12-bit arithmetic.

2.4 Microcoding the Am2901

The butterfly operations in Figures C-1 and C-2, the final rotations in (C-3) and (C-4), if they are used, and the DPCM calculations in (C-5) and (C-7) are carried out in the microprocessor system. Input and output buffering are accomplished by buffers as described in Sections 3.4, except that the variable-bit output requires some processor control. The entire system is pipelined, and the operations described in this section constitute the critical slowest step.

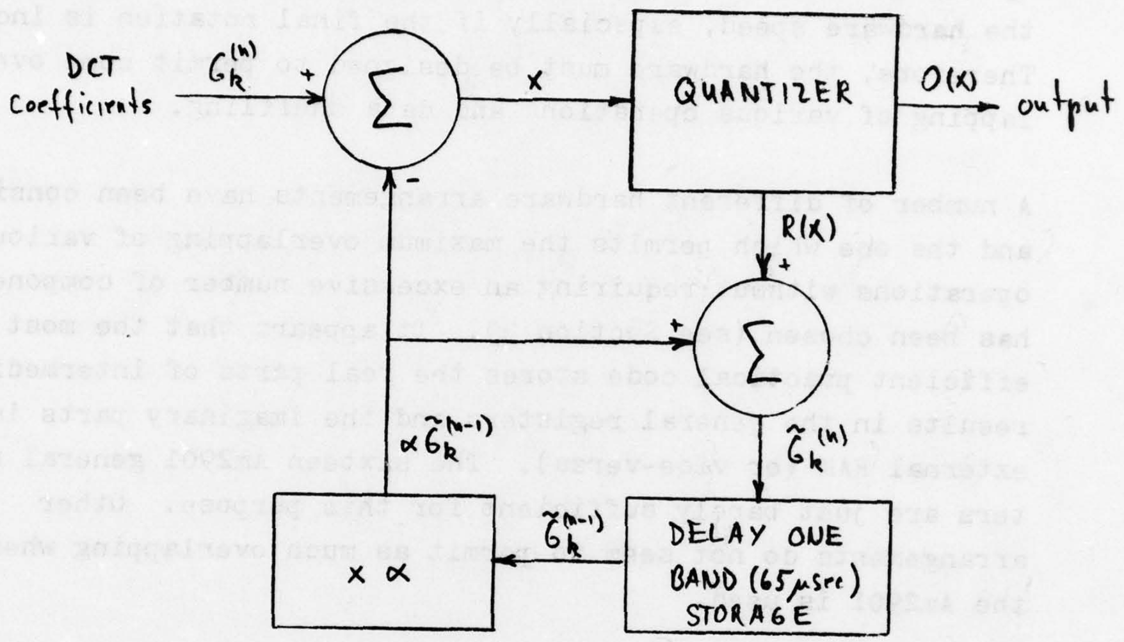


Figure C-4. Structure of the DPCM

Table C-2 gives the operation count for processor operations. If the computation rate is much slower than one operation per cycle, the cycle time will be less than the minimum permitted by the hardware speed, especially if the final rotation is included. Therefore, the hardware must be designed to permit some overlapping of various operations and data shuffling.

A number of different hardware arrangements have been considered, and the one which permits the maximum overlapping of various operations without requiring an excessive number of components has been chosen (see Section 3). It appears that the most efficient practical code stores the real parts of intermediate results in the general registers and the imaginary parts in an external RAM (or vice-versa). The sixteen Am2901 general registers are just barely sufficient for this purpose. Other arrangements do not seem to permit as much overlapping when the Am2901 is used.

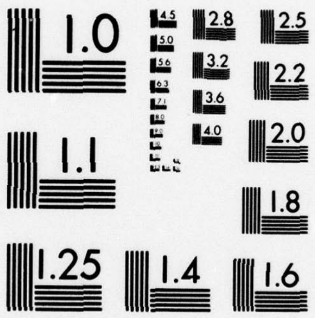
Twelve-bit twos complement arithmetic is used. This gives good results with negligible roundoff error (see Section 2.5), whereas eight-bit arithmetic would probably produce excessive round off error. Also, the quantization tables (Attachment 2) seem to require a 10-bit input.

The actual coding for a typical DCT operation, a "general butterfly", is shown in Figure C-5. The values of $x + w^{-2k}y$ and the conjugate of $x - w^{-2k}y$ are to be computed and stored with their real parts in the Am2901 general registers and their imaginary parts in the external RAM, as x and y were stored. It is assumed that $k \neq 4$, since $w^{-8} = \frac{1}{2}\sqrt{2}(1-i)$ gives a slightly simpler butterfly with $\theta = 45^{\circ}$. Notice that although multiplication takes two cycles, it is effectively pipelined.

AD-A072 917 DATA/WARE DEVELOPMENT INC SAN DIEGO CALIF F/G 9/2
ADVANCED DIGITAL TV SYSTEM.(U)
FEB 79 P J ERDELSKY, R V KEELE, G G MURRAY F33615-77-C-1198
UNCLASSIFIED AFAL-TR-79-1006 NL

4 OF 4
AD
A072917

END
DATE
FILMED
9-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

TABLE C-2

DCT/DPCM OPERATIONS

	Additions	Multiplications	Total
DCT	164	54	218
Final rotations	30	61	91
DPCM	64	32	96
Totals	258(228)	147(86)	405(314)
Time for each operation (nsec)			160(207)

Operation counts for the DCT-DPCM processor. Figures in parentheses exclude the final rotations.

b,d in general registers

c,e in RAM

$b \pm (d\cos\theta - e\sin\theta) \rightarrow$ general registers

$\pm c + (d\sin\theta + e\cos\theta) \rightarrow$ RAM

RAM → MIR	e → MIR
GR → MIR, MIR → PPR	d → MIR,
PSR → PIR, MIR → PPR	esinθ → PIR
PIR → Q, PPR → PIR, MIR → PPR	esinθ → Q, dcosθ → PIR
PIR -Q → Q, PPR → PIR	dcosθ - esinθ → Q, dsinθ → PIR
GR -Q → GR, RAM → MIR	b - Q → GR, e → MIR
GR + Q → GR, MIR → PPR,	b + Q → GR
PIR → Q, PPR → PIR	dsinθ → Q, ecosθ → PIR
Q + PIR → Q, RAM → PIR	Q + ecosθ → Q, c → PIR
PIR + Q → RIR (RAM → MIR)	c + Q → RIR (e → MIR)
-PIR + Q → RIR, RIR → RAM	-c +Q → RIR, c + Q → RAM
RIR → RAM(GR → MIR, MIR → PPR)	-c +Q → RAM, (d → MIR)

RIR:	Scratchpad RAM Input Register (Fig. C-10,P.296)
MIR:	Multiplier Input Register (Fig. C-11,P.297)
PPR:	Partial Product Register (Fig. C-11, P.297)

Coding for a "general butterfuly", where $\theta \neq 45^\circ$. Operations in parentheses are for the following butterfly, if any. The total number of cycles required is 12, with 2 overlapped with the following butterfly.

Figure C-5.

Coding such as this has also been done for other kinds of operations. The resulting operation counts are shown in Table C-3. The cycle time required to complete all operations in 65 μsec is 148 nsec or less, which is within the capabilities of the proposed hardware.

2.5 Accuracy

The only round off error in the DCT computations is that in products of the form x cosθ or x sinθ. (The error in the DPCM computations can be assumed to be included in the quantization error.) This error is assumed to be uniformly distributed over $[-1, 1]$, and all such errors are assumed to be independent. The variance of such an error is then 1/3. There is no roundoff error in additions or subtractions, because integer arithmetic is being used.

The number of such errors is large enough to make the central limit theorem applicable, so all variances can be added. What matters most for the purposes of this problem is the effect of such errors on the reconstructed inputs, under the assumption that the inverse DCT is exact, or at least much more accurate than the DCT.

An error in an intermediate result is an error in the discrete Fourier transform of some or all of the inputs, since that is what the intermediate result is. If the intermediate result lies at a "$\pm$" or "i" junction in Figure C-2, there is no roundoff error attributable to the butterfly. If it lies at a "4" or "-4" junction, it was computed as $x + \frac{1}{2}\sqrt{2}(1-i)y$ or something similar, which produces variances of $\frac{1}{3}$ in its real and imaginary parts. Otherwise, it was computed as $x + (\cos\psi + i\sin\psi)y$, or something similar, which produces variances of 2/3 in its real and imaginary parts. The effect of such an error on the reconstructed g_j is just an inverse discrete Fourier transform of the error. The sum of the variances of the resulting errors

TABLE C-3

DCT/DPCM PROGRAM CYCLES

Operation	Cycles in one Operation		Instances of Operation		Total Cycles
	Total	Overlap	Total	Overlap	
+ in column (1)	6	3	16	15	51
+,i	3	0	15	11	45
4,-4	9	1	7	4	59
general butterfly	12	2	10	8	104
DPCM - B_0, B_{16}	10	0	1	0	10
rotation, DPCM-B_1-B_{15}	14	3	15	14	168
					437

Estimate of the number of cycles required for the DCT and DPCM. The estimate is "conservative" (too large), because overlapping of cycles between consecutive operations of different kinds is not taken into account.

in the reconstructed g_j is, by the discrete Parseval's equation, just the variance of the error divided by the block length of the transform. The total variance of the errors in all the reconstructed g_j due to all such roundoff errors due to computations in Figure C-2 is calculated in Table C-4.

Since the errors are evenly distributed among the inputs, the variance of the error in each g_j is $(49/12)/32 = .1276$, which is negligible.

TABLE C-4

ESTIMATE OF VARIANCES

Block Length	Instances	Variance in Output: Each block	Variance in Output: Total	Variance in Reconstructed Input
2	16	0	0	0
4	8	0	0	0
8	4	8/3	32/3	4/3
16	2	40/3	80/3	5/3
32	1	104/3	104/3	13/12
Total				49/12

Calculation of sum of variances of the errors in the reconstructed g_j due to roundoff errors in the calculations in Figure C-2. Notice that Figure C-2 illustrates only half of the discrete Fourier transform outputs mentioned in the text; the other half are the conjugates of those shown.

SECTION III

BANDWIDTH REDUCTION SYSTEM HARDWARE

A microprocessor based design using the Advanced Micro Devices Am2901 bipolar microprocessor appears to offer the optimal approach to the digital implementation of the Bandwidth Reduction System. The Am2901 combined with a ROM (or RAM optional) microcontrol unit provides a flexible system whose performance cannot be matched by an MSI equivalent without sacrificing power and size. An MSI implementation of the Am2901 4-bit slice for example, could require 15-20 16-pin devices at a typical operating power of 3.6 watts. The Am2901 is a single 40 pin device with a typical power dissipation of 0.97 watts.

Seven basic elements comprise the microprocessor system which appears in Figure C-6. At the heart of the system is the microprocessor that performs the sum and difference calculations of the DCT and DPCM algorithms, stores temporary results in a 16 word dual port memory, and transfers data to different elements of the system. An additional 64 words of memory is provided by 2 64X9 RAMS to allow adequate storage for all 16 complex DCT coefficients and 32 DPCM results. A 12X12 multiplier supplies the high speed multiply capability that is imposed by real time processing and the quantizer produces the rounded and system output values of the DPCM algorithm. System I/O is accomplished through an input interface that digitizes and stores video data for processing and an output interface that converts parallel data received in a burst mode to a serial bit stream that is transmitted at one of four data rates, 200K bits/sec, 400K bits/sec, 800K bits/sec and 1600K bits/sec. DCT and DPCM algorithms will reside in the memory of the ROM (or RAM if desired for laboratory experimentation) based control unit that manages the operation of each system element.

These elements have been carefully designed and arranged to produce a system structure that can be programmed using a high degree of pipelining. Propagation delays through the various

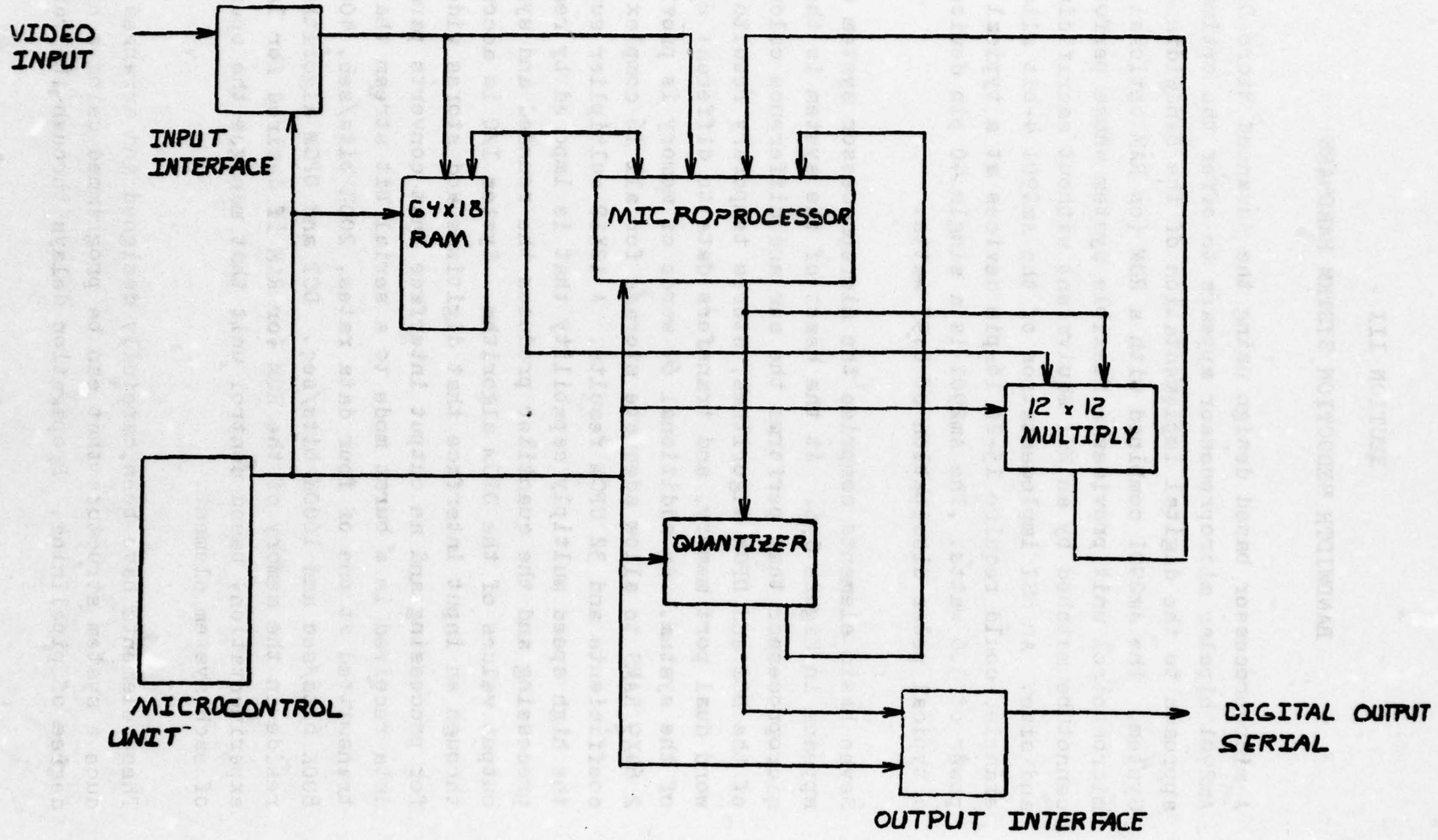


Figure C-6. BANDWIDTH REDUCTION SYSTEM BLOCK DIAGRAM system elements are matched using data latches where necessary and numerous data paths are available so that parallel data transfers are possible.

To realize the full potential of this pipeline architecture programs will be coded entirely in microcode using straight line programming. Using this method a maximum number of parallel operations can occur and the inefficiencies attributable to a hierarchy of code, program branches etc. are eliminated.

3.1 Microprocessor and RAM

The Am2901's surrounded by the other components that make up the microprocessor element appear in Figure C-7. Three Am2901's operate in parallel to give a 12-bit word length. A "look ahead carry" generator is included to speed up arithmetic operations.

The architecture of the Am2901 4-bit slice appears in Figure C-8. It's important features include a 16 word 2 port RAM with left or right shift inputs, a temporary storage Q register with left or right shift inputs, an ALU with several sources for operands, and a three state output buffer that can select either the ALU or the 2 port RAM for output. In addition the Am2901 provides the status outputs F=0, F_3, OVR and C_n+4, the generate and propagate outputs G, P for high speed "look ahead carry" operations and bi-directional ports for multi slice right and left shift operations.

Microinstructions for the Am2901 appear in Figure C-9. Instructions include 3 basic arithmetic operations and 5 logic operations that can be performed on several combinations of the A, B, D, and Q inputs. Note that both A-D-1 and D-A-1 can be performed which is not a common feature for a standard MSI ALU like the 74181.

Direct input data to the Am2901's (Figure C-8) is routed via a 4 to 1 multiplexer to a 12-bit input register. The input

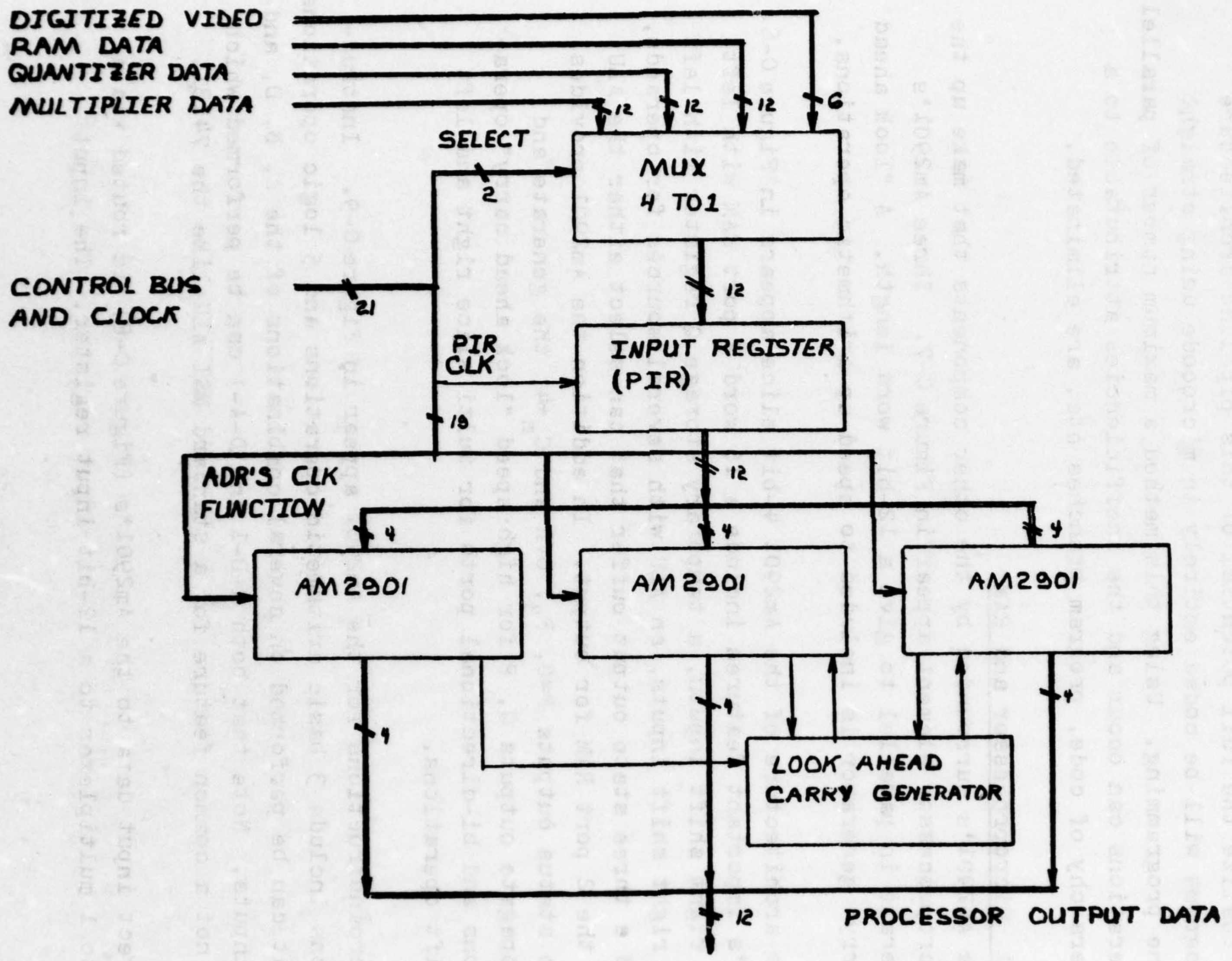
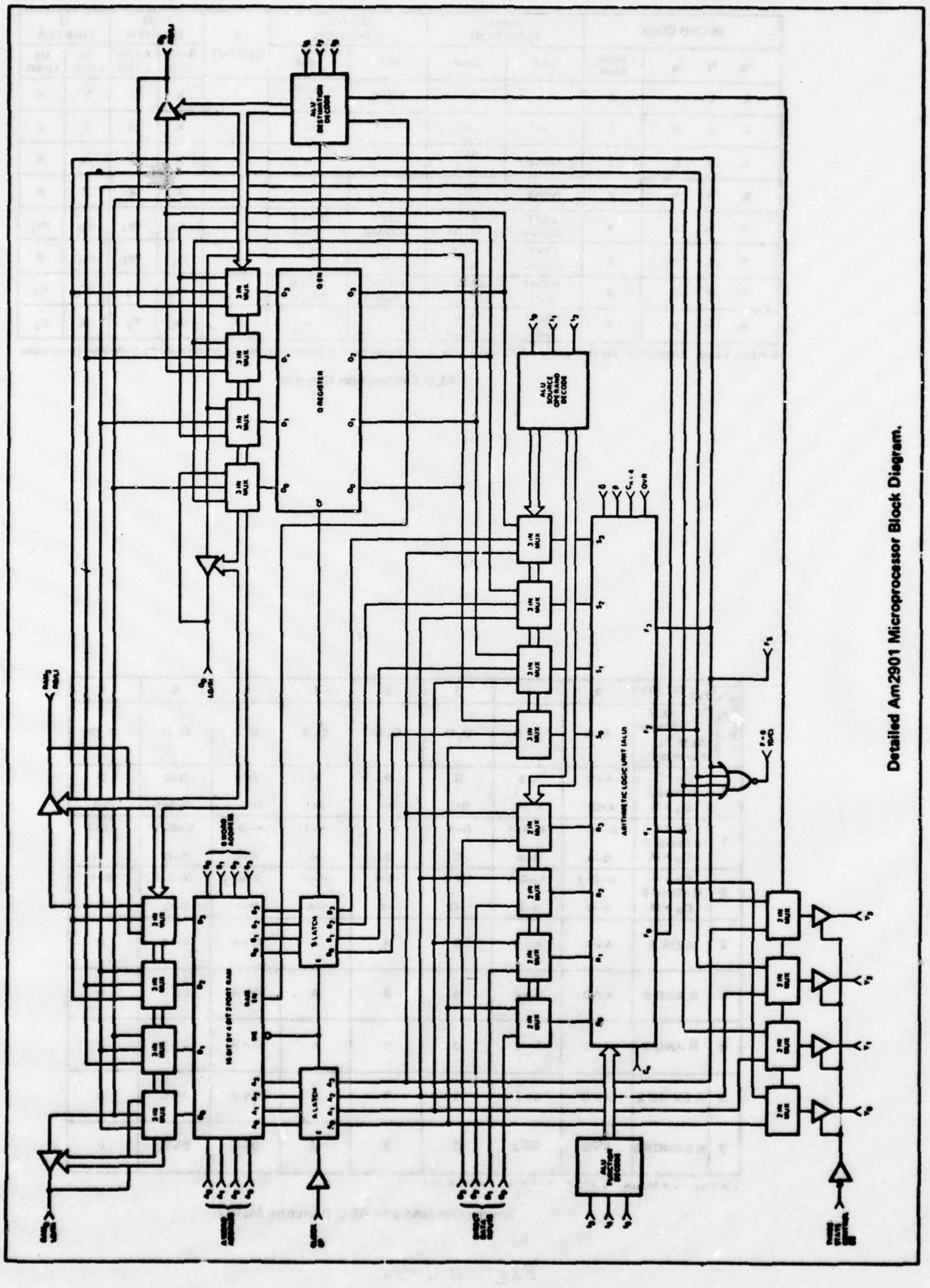


Figure C-7 BANDWIDTH REDUCTION SYSTEM MICRO-PROCESSOR ELEMENT



Detailed Am2901 Microprocessor Block Diagram.

Figure C-8.

MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I_8	I_7	I_6	Octal Code	Shift	Load	Shift	Load		RAM_0 LO/RI	RAM_3 LI/RO	Q_0 LO/RI	Q_3 LI/RO
L	L	L	0	–	–	NONE	ALU (F_i)	F	X	X	X	X
L	L	H	1	–	–	–	–	F	X	X	X	X
L	H	L	2	NONE	ALU (F_i)	–	–	A	X	X	X	X
L	H	H	3	NONE	ALU (F_i)	–	–	F	X	X	X	X
H	L	L	4	LEFT (DOWN)	ALU (F_{i+1})	LEFT (DOWN)	Q-REG (Q_{i+1})	F	F_0	IN_3	Q_0	IN_3
H	L	H	5	LEFT (DOWN)	ALU (F_{i+1})	–	–	F	F_0	IN_3	Q_0	X
H	H	L	6	RIGHT (UP)	ALU (F_{i-1})	RIGHT (UP)	Q-REG (Q_{i-1})	F	IN_0	F_3	IN_0	Q_3
H	H	H	7	RIGHT (UP)	ALU (F_{i-1})	–	–	F	IN_0	F_3	X	Q_3

X = Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state.

ALU Destination Control.

OCTAL I_{543}	I_{210} OCTAL / ALU Source / ALU Function	0	1	2	3	4	5	6	7
		A, Q	A, B	O, Q	O, B	O, A	D, A	D, Q	D, O
0	C_n = L R Plus S	A+Q	A+B	Q	B	A	D+A	D+Q	D
	C_n = H	A+Q+1	A+B+1	Q+1	B+1	A+1	D+A+1	D+Q+1	D+1
1	C_n = L S Minus R	Q−A−1	B−A−1	Q−1	B−1	A−1	A−D−1	Q−D−1	−D−1
	C_n = H	Q−A	B−A	Q	B	A	A−D	Q−D	−D
2	C_n = L R Minus S	A−Q−1	A−B−1	−Q−1	−B−1	−A−1	D−A−1	D−Q−1	D−1
	C_n = H	A−Q	A−B	−Q	−B	−A	D−A	D−Q	D
3	R OR S	$A \vee Q$	$A \vee B$	Q	B	A	$D \vee A$	$D \vee Q$	D
4	R AND S	$A \wedge Q$	$A \wedge B$	0	0	0	$D \wedge A$	$D \wedge Q$	0
5	$\bar{R}$ AND S	$\bar{A} \wedge Q$	$\bar{A} \wedge B$	Q	B	A	$\bar{D} \wedge A$	$\bar{D} \wedge Q$	0
6	R EX-OR S	$A \veebar Q$	$A \veebar B$	Q	B	A	$D \veebar A$	$D \veebar Q$	D
7	R EX-NOR S	$\overline{A \veebar Q}$	$\overline{A \veebar B}$	$\bar{Q}$	$\bar{B}$	$\bar{A}$	$\overline{D \veebar A}$	$\overline{D \veebar Q}$	$\bar{D}$

+ = Plus; − = Minus; $\vee$ = OR; $\wedge$ = AND; $\veebar$ = EX-OR

Source Operand and ALU Function Matrix.

Figure C-9.

register is located at the direct data inputs to assure that processing through the longest path (the Am2901) can be completed within a cycle time. The typical cycle time for this path is approximately 120 ns. Output data from the Am2901 goes directly to the other system elements.

The RAM which appears in Figure C-10 enhances the data storage capability of the Am2901. Data enters the RAM via a 2 input multiplexer to an input register which is required since two clock cycles are needed to compute and store data in RAM.

The RAM is organized, using 2 devices, as 64 words by 18 bits. While 18 bits is too large, 12 16X4 RAMS or 3 256X4 RAMS would be required to provide the equivalent storage of the 2 64X9 devices.

3.2 12X12 Multiplier

The 12X12 multiplier of Figure C-11 is implemented with eighteen Am2505 two's complement multipliers and five 4-bit "carry look ahead" adders. The multiplier is configured as two 6X12 multipliers with an adder to produce a final 12 bit truncated product. Propagation delays, through the multiplier, in this configuration are much less than those obtained by a straight forward single 12X12 configuration without adders.

The multiplier requires two cycles to generate a product. During cycle 1 the multiplicand enters the multiplier via a 2 to 1 multiplexer and is held in the input register. The second operand, the multiplier, is supplied by the control unit and held stable by the control word latch. During the second cycle the two partial products are held by the partial product register and the final product is computed.

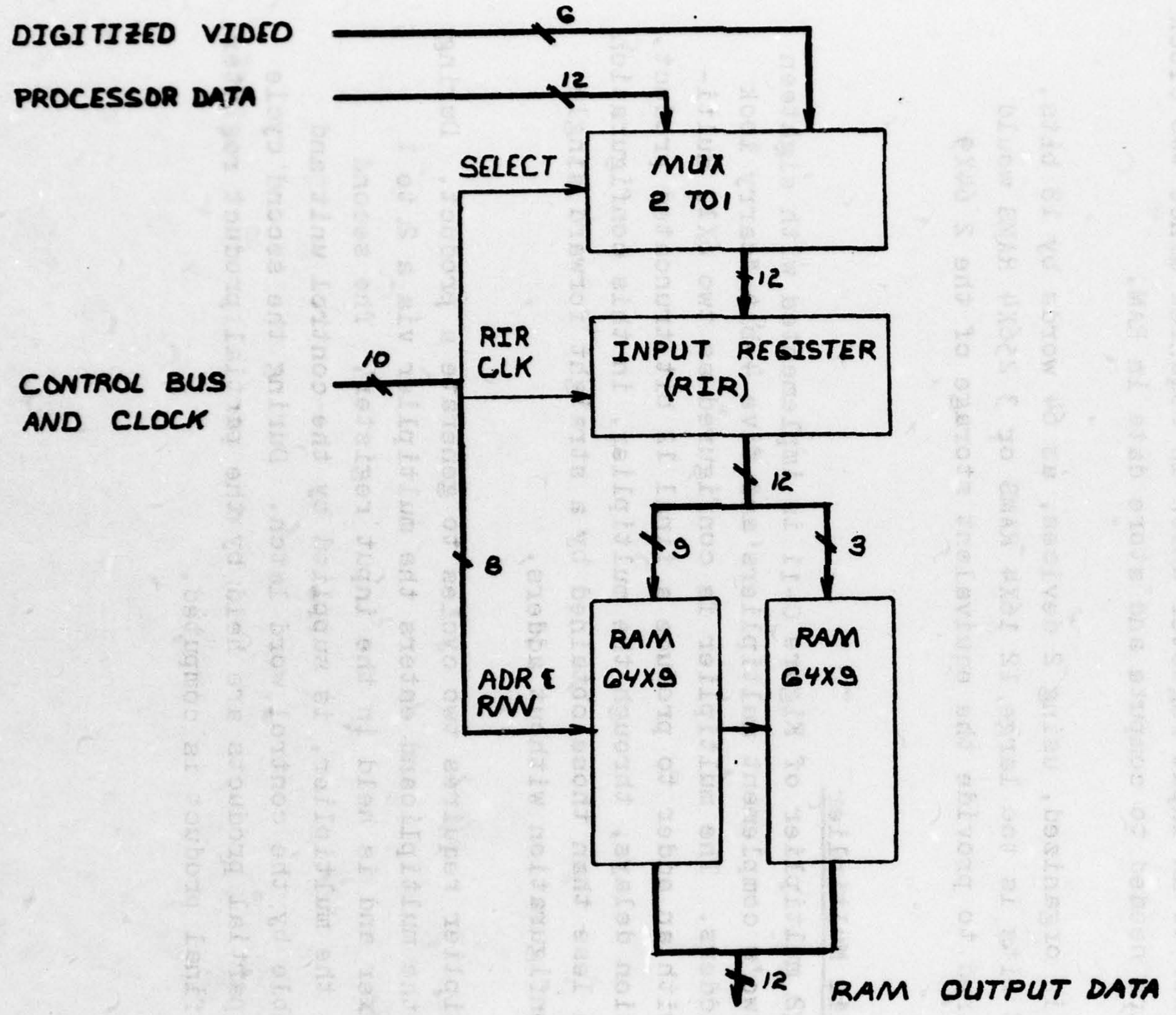


Figure C-10. BANDWIDTH REDUCTION SYSTEM DATA STORAGE ELEMENT

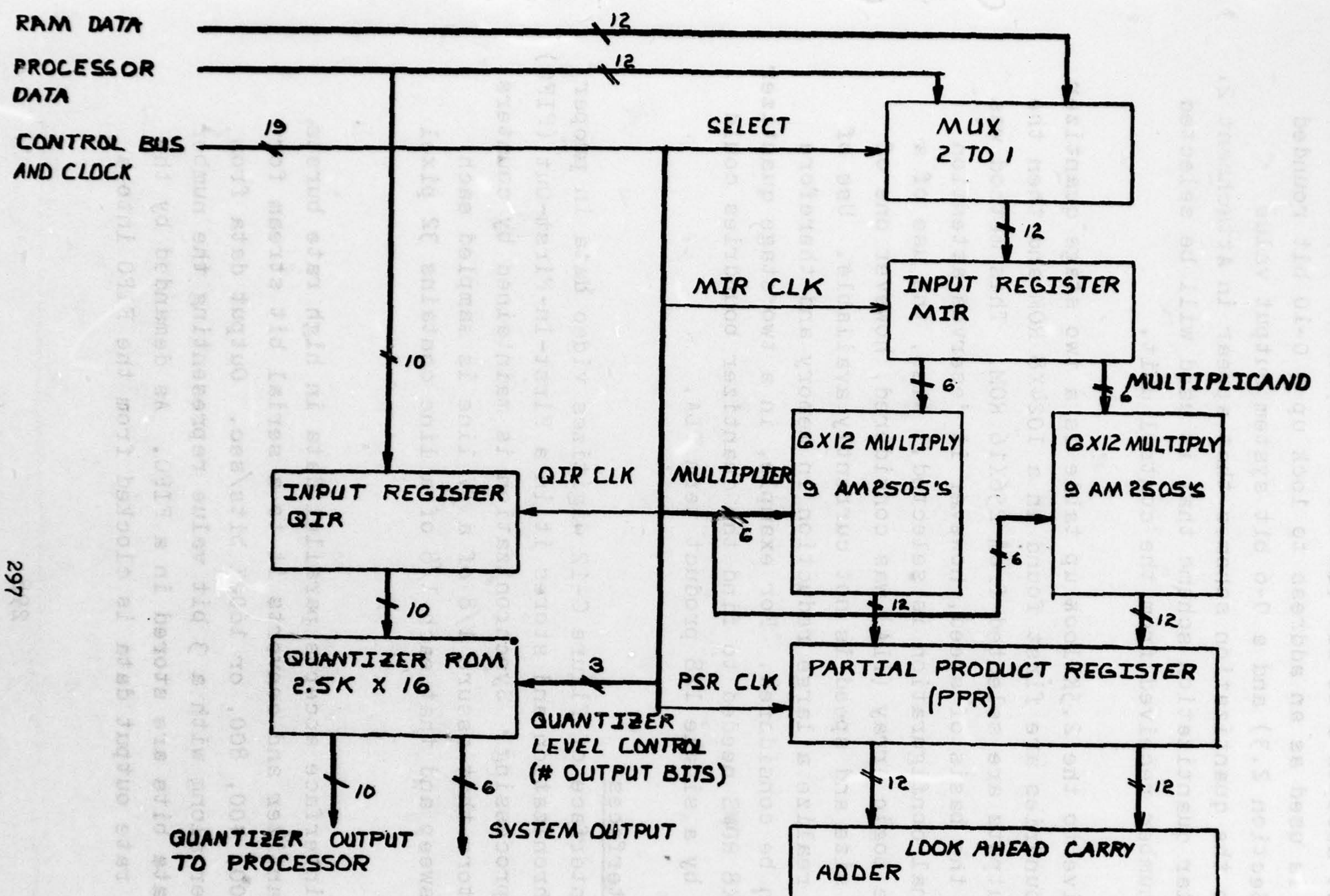


Figure C-11. BANDWIDTH REDUCTION SYSTEM QUANTIZER AND MULTIPLIER

3.3 Quantizer

The quantizer that appears in Figure C-11 is a 2.5KX16 ROM look up table that accepts a 10 bit (in two's complement representation) input which is used as an address to look up a 0-10 bit rounded value (see Section 2.3) and a 0-6 bit system output value according to the quantization schemes that appear in Attachment 2. The particular quantization scheme that is used will be selected by a 3 bit number received from the control unit.

An alternative to the 2.5K look up table is a two stage quantizer where the boundries are first found in a 1024X8 ROM and then the quantized outputs are selected from 256X16 ROM. This method was rejected on the basis of speed, however it deserves attention before a final configuration is selected. Also, the use of a programmable Logic Array (PLA) was considered, however one of the correct size and speed is not currently available. Use of a PLA would realize a large reduction in memory and therefore should again be considered. For example, in a two stage quantizer the two 512X8 ROMS needed to find the quantizer boundries could be replaced by a single 128 product term PLA.

3.4 I/O Interfaces

The input interface of Figure C-12 digitizes video data in proper system synchronization and stores it in a First-In-First-Out (FIFO) memory for processing. Synchronization is maintained by counters and comparators that assure 1/8 of a TV line is sampled each horizontal sweep and that each 1/8 of a line contains 32 pixel samples.

The output interface accepts parallel data in high rate bursts from the quantizer and converts it to a serial bit stream for output at 200, 400, 800, or 1600K bits/sec. Output data from the quantizer along with a 3 bit value representing the number of output data bits are stored in a FIFO. As demanded by the serial data rate output data is clocked from the FIFO into a

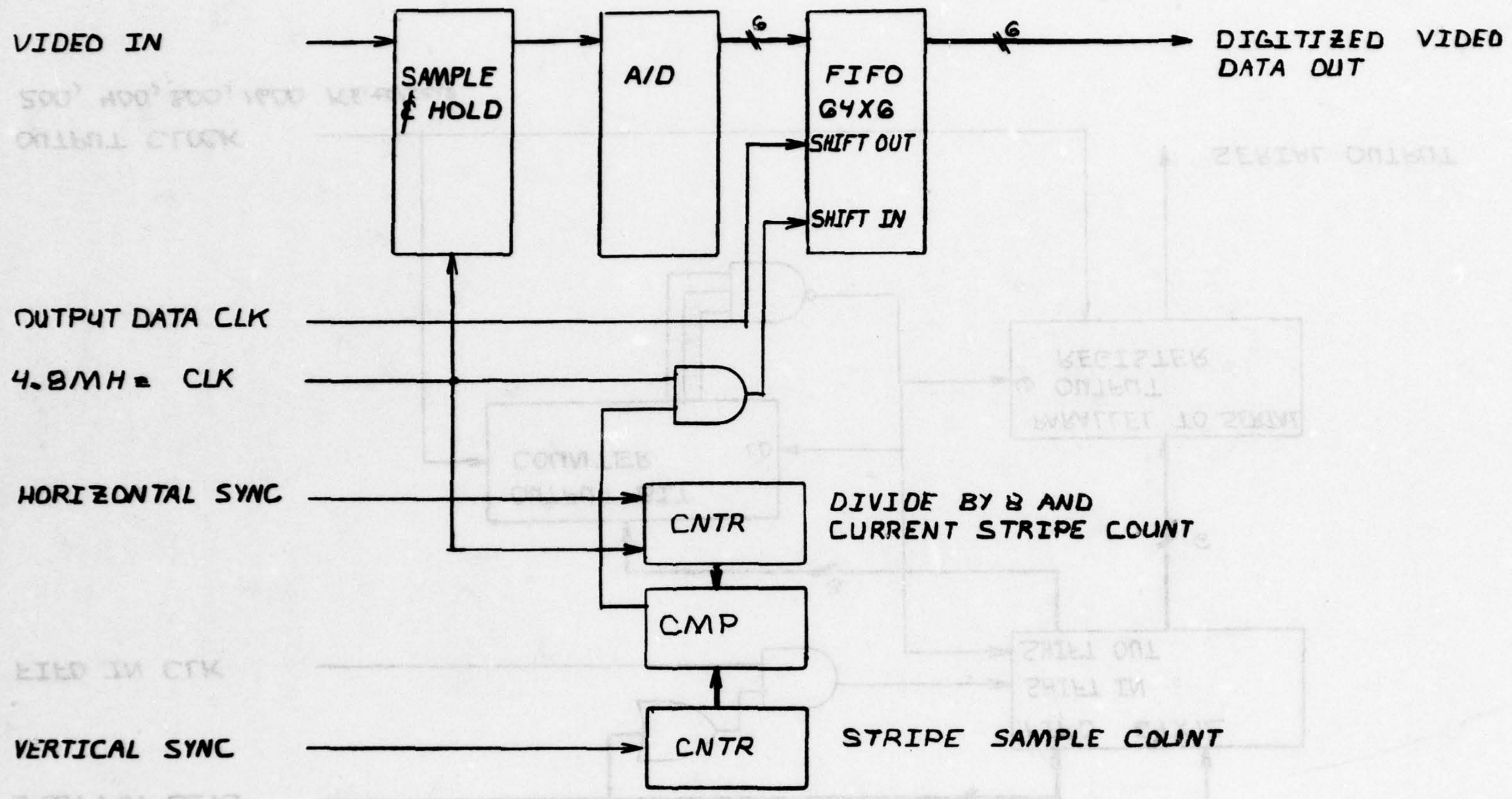


Figure C-12. BANDWIDTH REDUCTION SYSTEM INPUT INTERFACE

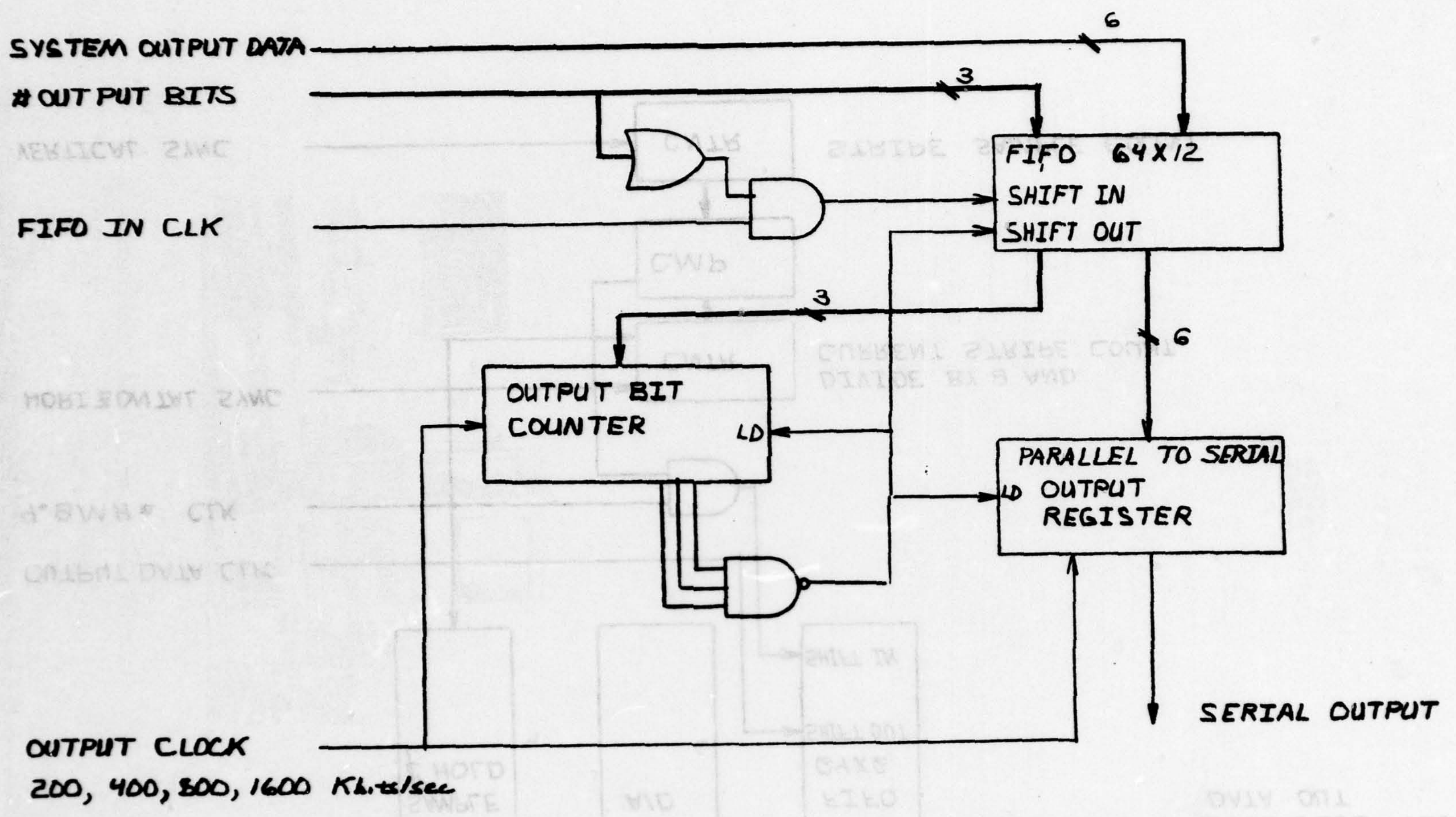


Figure C-13. BANDWIDTH REDUCTION SYSTEM OUTPUT INTERFACE parallel in serial out shift register. The bit count is loaded into a counter that is decremented each time a serial bit is clocked out. When the counter reaches zero the next set of parallel data is requested from the FIFO.

3.5 Micro Control Unit

As previously stated the micro control unit of Figure C-14 is a pipelined memory based system that controls all system elements. Algorithms implemented with this system will be contained in the microcontrol memory.

The control memory may be either ROM for production applications or RAM to facilitate laboratory investigations. RAM memory can be loaded by use of the external (EXT) controls and data lines indicated.

A 12 bit counter provides addressing to the control memory during system operation and the system clock increments the address and synchronizes the system. A single bit from the control memory via the control word registers signals the end of a micro program and returns the system to the start of the program.

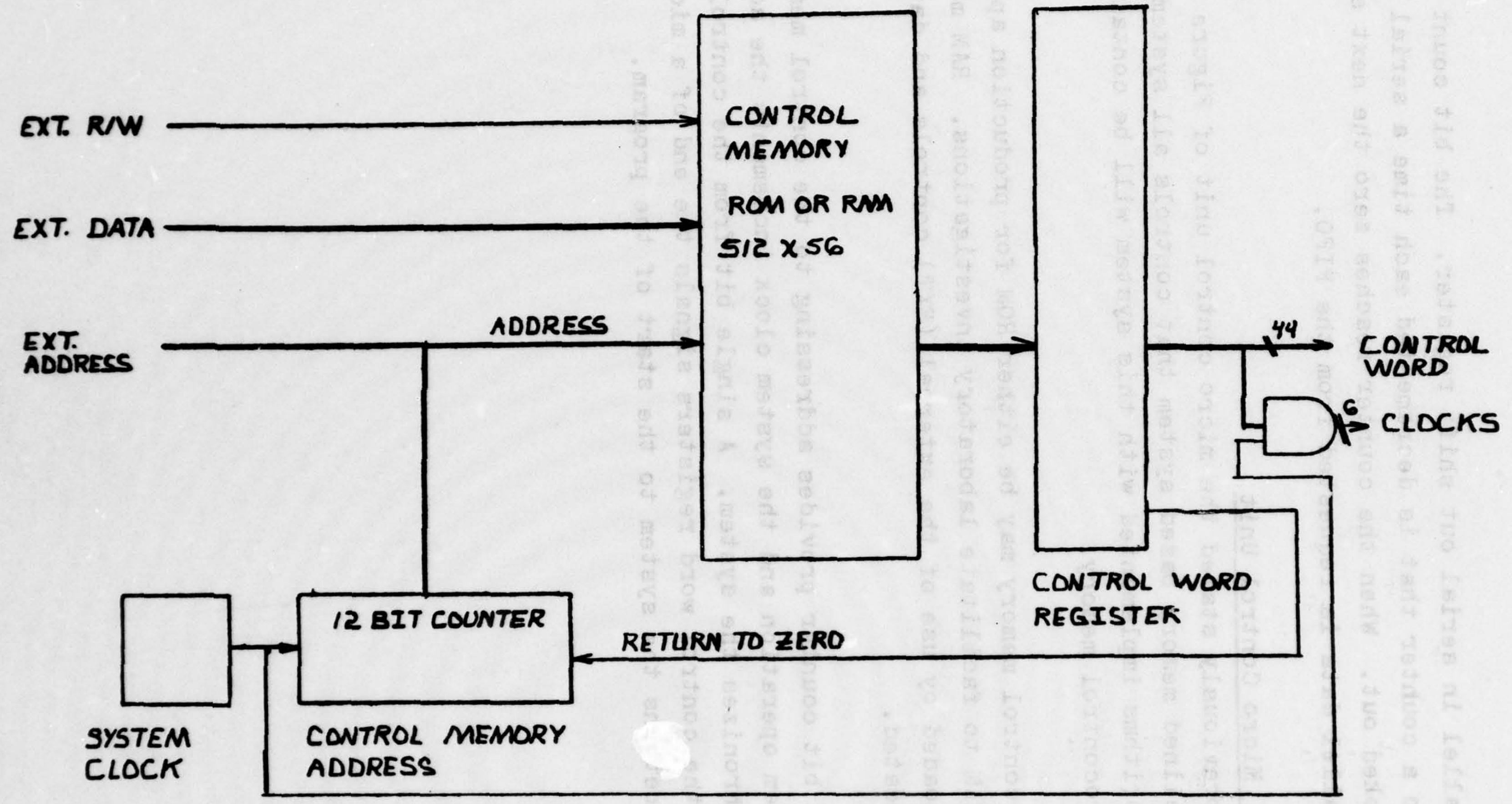


Figure C-14. BANDWIDTH REDUCTION SYSTEM MICROCONTROL UNIT

ATTACHMENT 1 TO APPENDIX C

DIRECT CALCULATION OF FFT'S WITH REAL INPUTS

Suppose $a_0, a_1, \ldots, a_{4N-1}$ are real, $w=\exp(\frac{1}{2}\pi i/N)$ and

$$A_k = \sum_{j=0}^{4N-1} a_j w^{-jk}, \quad k = 0, 1, \ldots, N-1$$

Then since $w^{4N} = 1$

$$A_{4N-k} = \sum_{j=0}^{4N-1} a_j w^{-4jN} w^{jk}$$

$$= \sum_{j=0}^{4N-1} a_j w^{jk} = \bar{A}_k$$

Since the entire FFT output can easily be reconstructed from this rule and the values of $A_0, A_1, \ldots, A_{2N}$, only these values will be computed. It is easily shown that A_0 and A_{2N} are real, so these values can be conveniently stored in 4N cells as follows:

A_0, A_{2N}
$\mathrm{Re}A_1, \mathrm{Im}A_1$
$\mathrm{Re}A_2, \mathrm{Im}A_2$
* * *
$\mathrm{Re}A_{2N-1}, \mathrm{Im}A_{2N-1}$

The usual FFT decimation is

$$A_k = B_k + w^{-k}C_k, \quad k = 0, 1, \ldots, 2N-1$$

$$A_{k+2N} = B_k - w^{-k} C_k, \quad k = 0, 1, \ldots, 2N-1$$

$$B_k = \sum_{j=0}^{2N-1} a_{2j} w^{-2jk}, \quad k = 0, 1, \ldots, 2N-1$$

$$C_k = \sum_{j=0}^{2N-1} a_{2j+1} w^{-2jk}, \quad k = 0, 1, \ldots, 2N-1$$

Since B_k and C_k are also DFT's with real inputs, B_0, B_N, C_0 and C_N are real and

$$B_{2N-k} = \bar{B}_k, \quad C_{2N-k} = \bar{C}_k$$

Then since $w^{2N} = -1$,

$$A_{2N-k} = B_{2N-k} + w^k w^{-2N} C_{2N-k}$$

$$= \bar{B}_k - w^k \bar{C}_k$$

$$\bar{A}_{2N-k} = B_k - w^{-k} C_k$$

and A_0, A_1, ..., A_{2N} can be expressed in terms of B_0, B_1, ..., B_N, C_0, C_1, ..., C_N by

$$A_0 = B_0 + C_0, \; A_{2N} = B_0 - C_0$$

$$A_N = B_N - iC_N$$

$$A_k = B_k + w^{-k} C_k, \quad k = 1, 2, \ldots, N-1$$

$$\bar{A}_{2N-k} = B_k - w^{-k} C_k, \quad k = 1, 2, \ldots, N-1$$

If N is even, the same procedure can be used to compute B_k and C_k. If $N = 2^n$, the process can be repeated until trivial FFT's of block length 1 are reached. It can be shown that this process requires $2N(3n+1)+4$ real additions and $2N(2n-3)+6$ real multiplications, respectively, when $n \geq 1$.

The usual method of channel separation requires the same number of multiplications but slightly more additions and substantially more storage, since two transforms must be done at the same time.

ATTACHMENT 2 TO APPENDIX C

QUANTIZER VALUES

Quantizer Positive Values for 64 levels

Quantizer Bound	Quantizer Value	Transmitted Bits
0		
	1	000000
2		
	3	000001
4		
	5	000010
6		
	7	000011
8		
	9	000100
10		
	11	000101
13		
	14	000110
15		
	16	000111
18		
	19	001000
20		
	21	001001
23		
	24	001010
26		
	27	001011
29		
	30	001100
32		
	33	001101
35		
	37	001110
39		
	41	001111
43		
	45	010000
47		
	49	010001
51		
	53	010010
55		
	57	010011
60		
	63	010100
66		
	69	010101
72		
	75	010110
78		
	81	010111
85		
	89	011000
94		
	98	011001
103		
	108	011010
114		
	121	011011
128		
	137	011100
146		
	158	011101
171		
	190	011110
213		
	275	011111

DPCM Quantizer Values

Quantizer Positive Values For 32 Levels

Quantizer Bound	Quantizer Value	Transmitted Bits
0	2	00000
4	6	00001
8	10	00010
13	15	00011
18	20	00100
23	26	00101
29	32	00110
35	39	00111
43	47	01000
51	55	01001
60	66	01010
72	78	01011
85	94	01100
103	115	01101
128	147	01110
171	233	01111

DPCM Quantizer Values - (Continued)

Quantizer Positive Values For 16 Levels

Quantizer Bound	Quantizer Value	Transmitted Bits
0		
	4	0000
8		
	13	0001
18		
	23	0010
29		
	36	0011
43		
	51	0100
60		
	72	0101
85		
	104	0110
128		
	190	0111

Quantizer Positive Values For 8 Levels

Quantizer Bound	Quantizer Value	Transmitted Bits
0		
	9	000
18		
	30	001
43		
	62	010
85		
	147	011

DPCM Quantizer Values - (Continued)

Quantizer Positive Values For 4 Levels

Quantizer Bound	Quantizer Value	Transmitted Bits
0		
	19	00
43		
	105	01

Quantizer Positive Values For 2 Levels

Quantizer Bound	Quantizer Value	Transmitted Bits
0		
	62	0

DPCM Quantizer Values - (Continued)

APPENDIX D

Presented at the 1977
National Telecommunications Conference

DIGITAL TV MICROPROCESSOR SYSTEM

Ronald A. Belt
USAF Avionics Lab
Wright-Patterson AFB

Richard V. Keele G. Graham Murray
Data/Ware Development
San Diego, California

Abstract

Employing a sophisticated data compression algorithm, the laboratory version of a digital TV system which exploits the power of off-the-shelf microprocessor chips has been designed and placed in operation. TV images are treated as 256 lines of 256 pixels (picture elements), the TV camera video being converted to 6 bits of accuracy. Frame slow down by a factor of 8 combined with the Discrete Cosine Transform (DCT) along a TV line and Differential Pulse Code Modulation (DPCM) line to line can reduce the data rate to as low as 200 kbps. At the receiving site the inverse transformations are performed to recover the image, which is stabilized in a digital Frame Store Memory (FSM) during conversion back to analog for display on the TV monitor.

Introduction

Under USAF sponsorship* a laboratory system has been developed for the evaluation of digital TV transmission systems, as illustrated in Figure 1. Such a system could be employed for the communication of TV signals from a Remotely Piloted Vehicle or Spacecraft to a ground station. Rather than assembling the rather limited configuration of Figure 1, a much more powerful, general-purpose system was placed in operation. As shown in Figure 2, it replaces the modems with the UNIBUS of the PDP-11 minicomputer--at the same time permitting the minicomputer to have control over all the subsystems and the mode of operation.

The two key elements are the microprocessors which are 12-bit units assembled from the Schottky TTL Am 2901 4-bit-slice microprocessor chips. Capable of performing any arithmetic or logical operation (including multiply) in 150 ns, each Model 1240 microprocessor runs under microprogram control from a 48-bit wide RAM microstore to which the PDP-11 has direct access.

Because the system is entirely microcoded, it can serve to study any of a wide variety of compression algorithms. The initial one demonstrated is that due to Habibi, sometimes called a hybrid technique because it is a combination DCT/DPCM. It has been shown by simulation to give performance close to the optimum of the Karhunen-Loeve transform. It should be noted that the system can just as readily enhance images as compress them. The microprocessors can be programmed to perform any desired 1-D or 2-D enhancement algorithm, the system having been designed to permit this. Medical image processing is a definite possibility with this configuration.

*Contract F33601-76-90528

System Operation

It was convenient to place all the system elements on the UNIBUS to permit flexibility of operation. In the PDP-11 any two devices attached to the UNIBUS (including the CPU and high-speed memory) can communicate, provided that one acts as Bus Master and the other as Bus Slave. The CPU is always a Master and memory a Slave. The Model 1240 system makes the TV Camera a Slave, but the Frame Store Memory and the microprocessors can be either Masters or Slaves.

The UNIBUS permits a maximum rate of transfer of 2.5 megawords (5.0 megabytes) per sec. This is sufficient to allow sending the 256 pixels of a TV line from the camera to any destination in the available 63.5 microseconds. In particular, an entire field of 256x256 pixels can be transferred to the 65,536 byte FSM.

One mode which is quite important is that in which a "stripe" 32 pixels wide and 256 lines deep is transmitted during a field time. This is an 8 to 1 field slow down which gives acceptable results at the ground station in typical airborne applications. The stripes are processed from left to right, each stripe being updated at a 7.5 times a second rate.

When the system is in the stripe mode, 32 pixels from each line are sent to the first microprocessor which performs the DCT/DPCM computation on the entire stripe. Note that a 32-point DCT computation, followed by the DPCM, must be executed by the microprocessor in 63.5 microseconds or less. Employing an algorithm for the DCT superior to any published which is due to P. J. Erdelsky and G. G. Murray of Data/Ware, the DCT can be computed with 194 additions and 115 multiplications. With its pipelined architecture and fast 150 ns microinstruction time, the Model 1240 microprocessor is able to perform these computations in the allocated time.

The second microprocessor, which simulates the ground station, must carry out the inverse transformations in order to recover the image. The data compression achieved results from the DPCM process, which is the last step at the transmitting site. At the ground station the inverse DPCM is the first operation.

After the DCT step during encoding, the DPCM compares the frequency domain coefficients of one TV line segment with those on the succeeding line. A difference is formed for each coefficient which serves as input to a quantizer table. Depending upon the amount of compression desired, the code word from the quantizer can vary from 0 bits to perhaps 4 bits. One output from the quantizer is a rounded value of the difference, which when added to the value of the coefficient on the preceding line produces the estimate for the present line. The second output is the code word, which is in a one-to-one relationship with the rounded (quantized) value and is transmitted to the ground station.

Depending upon the contents of the quantizer, the overall bit rate can vary from 1,600,000 bps to 200,000 bps, the latter corresponding to less than 0.5 bits per pixel. There is of course noticeable degradation at lower bit rates, but the quality is adequate for many applications where fine detail is not needed.

As implemented, there are 312 clock pulses during each TV line -- 256 during the active time of 52.1 microseconds and 56 during the 11.4 microsecond retrace time. When in the stripe mode, the second microprocessor must update the FSM with 32 new pixels during each line time. This can conveniently be carried out during retrace time when it will not disturb the readout of the FSM as it updates the TV monitor.

Compression Algorithm

With the hybrid DCT/DPCM compression scheme, it is the DCT step which by far is the more difficult. A transform on 32 pixels must be carried out in less than 64 microseconds. In addition, formulations of the DCT computational algorithm in the literature suggest in essence that a double-length Fast Fourier Transform (FFT) be performed. Let g_0, g_1, ..., g_{31} be the pixel grey-scale values from one stripe of a TV line. The DCT is defined as

$$G_k = 2\sum_{j=0}^{31} g_j \cos\left[(j+\tfrac{1}{2})k\theta\right] \qquad (1)$$

$$k=0,\ 1,\ \ldots,\ 31$$

where $\theta = 2\pi/64$. By combining complex conjugate terms,

$$G_k = w^{-\frac{1}{2}k}\sum_{j=0}^{63} a_j\, w^{-jk} \qquad (2)$$

where $w = e^{i\theta}$, i is the square root of -1, and

$$a_j = \begin{cases} g_j & \text{if } j \leq 31 \\ g_{63-j} & \text{if } j \geq 32 \end{cases}$$

The literature suggests this formulation: Let

$$A_k = \sum_{j=0}^{63} a_j\, w^{-jk} \qquad (3)$$

Then

$$G_k = w^{-\frac{1}{2}k} A_k \qquad (4)$$

where this last step is referred to as the final rotation. Note that equation (3) is the double-length FFT referred to above.

The new algorthim for the DCT performs the computation in two steps also. The first is a single-length, FFT-like computation in which quantities B_k are derived. This is illustrated in Figure 3, which identifies the four types of "butterfly" calculations required. A rotation is next carried out. After the rotation step, the G_k are available for input to the DPCM computation.

The DPCM step involves DCT coefficients from successive lines within a stripe. The notation is that $G_k^{(n)}$ is the kth coefficient for the nth line. A difference, x, is formed between the present coefficient and the corresponding one from the preceding line -- attenuated by a factor α between 0 and 1.

The difference, x, is used as the input to the quantizer, whose output depends upon x and upon k (the frequency of the coefficient). O(x) is the code word sent to the receiving site, and R(x) is the rounded value of the difference. The smoothed estimate of the kth coefficient is given by

$$\overline{G}_k^{(n)} = R(x) + \alpha\overline{G}_k^{(n-1)} \qquad (5)$$

where

$$x = G_k^{(n)} - \alpha\overline{G}_k^{(n-1)} \qquad (6)$$

The starting value of $\overline{G}_k^{(n-1)}$ is not important because α is less than 1 so that the initial value is attenuated to zero eventually.

System Implementation

Each major subsystem is interfaced to the UNIBUS via a plug-in card so that the various key registers of the subsystem appear as high-speed (core) memory locations to the PDP-11 CPU. This means that the PDP-11 can directly modify the control store contents within each microprocessor and interrogate or modify various registers. Thus the microprocessor is controlled not by a front panel but rather by an operating system within the PDP-11 computer. Additional registers within the interface card itself establish the overall mode of operation. Each microprocessor can be set up to act as Bus Master or Bus Slave. Typically the first microprocessor will act as Master in fetching pixels from the TV camera and in sending DCT/DPCM "coefficients" to the second microprocessor.

This requires the second microprocessor to be commanded to accept coefficients as a Slave but to send the recovered pixels to the FSM as a Master.

Another possibility is for the first microprocessor to send the coefficients to the core memory of the PDP-11, from which the second microprocessor can fetch them. This is convenient for 2-D work in which the first microprocessor performs the horizontal DCT, Fourier, Hadamard, or Haar transform, and the second microprocessor performs the vertical transform. The coefficients can now be manipulated by the PDP-11 either for enhancement or for redundancy reduction. This would be followed by the inverse transforms. Implied in the above is the ability of the microprocessor interface card to carry out "corner-turned" addressing of core memory when required for 2-D work.

Because the PDP-11 has access to the interface cards, it controls completely the mode of operation. The end of each field time is signalled to the PDP-11 as an interrupt. During vertical retrace time the CPU can set up the desired processing for the next field. For example, the CPU must change the stripe by incrementing various registers in the interface cards.

The FSM is regarded by the PDP-11 as a 32,768 word memory to which it has access as 8 pages, each of 4,096 words. This makes it quite convenient for the CPU to interrogate and manipulate pixels within the FSM. Conversely, it was mentioned above that the microprocessors can access the PDP-11 core memory, which greatly reduces the need for transferring blocks of data within the system.

As previously noted, the system can capture an entire TV field rather than operating in the stripe mode. This would permit avoiding the problem of edge effects from assembling 8 stripes into a single picture. Also, provision has been made for adding a second FSM. This would simulate a mode of operation in which the transmitting site seizes a field before compressing it.

In an actual application, the transmitting site would have to send a synchronizing signal. For laboratory work the synchronization is provided by a Fairchild 3262A sync generator chip whose outputs are fed to the camera and monitor. Through the use of FIFO memories for both input and output, the microprocessors can run independently from the synchronizing signals. When the input FIFO is empty or the output FIFO full, the microprocessor suspends operation.

Although it clearly results in a processing slow-down, it is possible to operate the system with a single microprocessor which will both compress the data and expand it. It is necessary to bring the field directly from the camera to the FSM. This data is then read out to the microprocessor for compression, followed by expansion. The processed pixels are then stored back in the FSM for viewing. When done dynamically with a single FSM, the TV monitor alternately shows the original pixels and the processed pixels unless the system has two FSM's. In the latter case the original TV images and the processed images could be viewed side by side.

Model 1240 Microprocessor

The microprocessor has been carefully designed for optimum performance in signal processing applications, especially those involving the Fourier Transform or the DCT. The three Am 2901 chips facilitate addition, subtraction, and logical operations. Figure 4 shows the architecture of the Model 1240, revealing a separate multiplier array, RAM Scratch Pad Memory of 64 words, and Quantizer Tables for the DPCM computation. Within the 48-bit microinstruction is a 12-bit operand field which can store sine/cosine values needed for the transform.

A pipeline architecture permits the execution of microinstructions in 150 ns with literally no overhead. For example, the general butterfly computation of the FFT requires 4 multiplies and 6 additions, exclusive of data fetches, stores, etc. The Model 1240 is able to perform the butterfly in exactly 10 microinstructions, including all fetches and stores. Parallel operations along with pipelining permits this. The coding is all in line in order to avoid wasted time on looping, testing, etc. The organization within the Am 2901 chip itself is very conducive to efficient computing. The availability of the 16 general registers plus the Q register is quite important. These are supplemented by the 64 words of RAM Scratch Pad Memory which is organized so that the Model 1240 never has to wait to fetch words or to store them. Fields within the 48-bit microinstruction are able to anticipate the demand for data and the need to store it so that memory operations are always overlapped with productive arithmetic ones.

In the organization of the Am 2901 two operands (either two general registers or a general register and externally supplied word) are combined with the result being stored in a general register. A word can be outputted in parallel with the preceding if desired. From Figure 4, the D register is the means of supplying the external word from any of several sources, and it can be seen that the Am 2901 output can be directed to any of several destinations.

To assist the programmer in writing code, a source language has been defined for input to a cross-assembler written for the PDP-11. Each line of source language generates at most one microinstruction. Lines typically contain several phrases, some examples of which are as follows: r op s → dest, S → Sn, D → Tn, Sn → D, nnnn X M, nnnn → D, JUMP. The interpretations are: combine the operands r and s (which may be in general registers), load the contents of register S into Scratch Pad Memory location n, load the contents of the D register into Quantizer Table location n, load Scratch Pad Memory location n into the D register, multiply the contents of register M by the operand field within the microinstruction, load the operand field into the

D register, and jump to either 0000 or the breakpoint.

In the phrase, "r op s", op refers to one of the Am 2901 operations: +, -, OR, AND, CAND, XOR or NXOR; "dest" refers to one of the destinations: R_n, Q, R_n SHIFTED, S, T, M, LOUT or ROUT; and r and s are one of the permitted combinations.

r	s
Ø	Q
Ø	R_n
D	Ø
D	Q
D	R_n
R_n	Q
R_m	R_n

R_n is one of the Am 2901's general registers, Q is the extension register, S is the Scratch Pad Register, T is the input Register to the Quantizer Table, M holds the multiplier, and LOUT and ROUT are the two output FIFO's (left and right).

Source language for the Model 1240 Micro-assembler is in ASCII format, typically from paper tape. Nulls, line feeds and rubouts are ignored, and the carriage return is used as a line terminator.

Each line of source language generates at most one microinstruction, and must not be any longer than 46 characters (not counting the carriage return that terminates it). Phrases, as illustrated above, are made up of "items", such as R_n, Q, D, OR, AND, etc., from a list of permitted items. Items must be in a particular order, but phrases may appear in any order.

Items should be separated by one or more spaces, but spaces are not required before or after non-alphanumeric characters. Phrases may be separated by spaces and/or commas. (Spaces and commas are generally ignored except as item terminators.)

Almost any ASCII string enclosed in parentheses is treated as a comment. It will appear in the assembler listing but will not affect the assembly otherwise. Comments may be placed anywhere except inside items. A comment at the end of a line does not need a right parenthesis, and comments may not contain carriage returns or right parentheses.

The microinstruction, consisting of 48 bits, can be loaded from the host minicomputer as three 16-bit words. Microstore is organized in the laboratory model as up to four cards, each of 1,024 microinstructions. As remarked above, however, the total DCT/DPCM program requires less than 512 microinstructions.

Each microinstruction is organized into 14 fields, numbered in octal. Fields consists of the following:

01	Causes writing to Scratch Pad
02	Scratch Pad Memory Address
03	Not used
04	Am 2901 A address
05	Am 2901 B address
06	Carry In
07	Am 2901 function
10	Control of loading D Reg
11	Am 2901 Source Operand
12	Am 2901 Destination Control
13	Control of Am 2901 output
14	Loads Multiplier from RAM
15	Selects Quantization Table
16	12-bit Operand

With the above logical orgainzation the programmer can arrange so that operands are always available when needed, either from Scratch Pad, general registers, or the input FIFO's. Furthermore, addition and multiplication can be overlapped. Field 16 provides a 12-bit twos complement number by which the contents of the M register are multiplied. Multiplication is performed in two stages -- first two partial products are formed by carrying out two 6x12 multiplications in parallel and then on the next clock cycle the two partial products are added. The sum is available for loading into the D register. Alternatively, Field 16 can be loaded directly into the D register.

Future Directions

Although it has been pointed out that the configuration of Figure 2, as well as the modified one-microprocessor system, would permit conducting many types of imagery studies, including enhancement, the original impetus for the project was developing low-cost TV compression hardware for Remotely Piloted Vehicles. Therefore, emphasis was placed on the utilization of off-the-shelf LSI circuitry. This goal has been successfully achieved although the hardware has not been repackaged in hybrid form to achieve the desired small size.

There is one problem which has yet to be addressed -- the power consumption of Schottky TTL. It is in this respect that the present design has a deficiency since it requires around 60 watts. What is needed is a CMOS (preferably SOS) version of the Am 2901. Several companies appear to be active in this field fortunately. Thus the design would have to be modified from TTL to CMOS, which would not be a major undertaking. At the same time it would be desirable to reduce the word length of the microprocessor, at least the airborne one, from 12 bits to 8 bits, which would reflect into further hardware and power savings.

One of the important achievements in the present design was the discovery of a totally new DCT algorithm, much more efficient than those previously employed. It appears, however, that there exist possible further improvements to the algorithm which would reduce the required computations without much effect on image quality. This too would have a favorable impact on the hardware.

Summary

It has been shown that it is feasible to combine 4-bit Schottky TTL microprocessor chips into a 12-bit processor capable of compressing TV signals sufficiently to achieve data rates as low as 200 kbps. The algorithm initially demonstrated is the Discrete Cosine Transform followed by Differential Pulse Code Modulation. A laboratory model in which the two microprocessors corresponding to the transmitting and receiving sites are controlled by a PDP-11 minicomputer has been designed and placed in operation. See Figure 5 which shows the imagery system mounted in a relay rack. With this laboratory system it is possible to simulate a complete imagery transmission link, whether it is one using the DCT/DPCM algorithm or one using another compression technique. A very powerful algorithm for the DCT has been demonstrated. For the ultimate in low-power consumption the present TTL design would be replaced with CMOS. Based upon new developments in CMOS LSI devices, including 4-bit slice architectures, this appears quite practical.

References

1. Habibi, A., "Hybrid Coding of Pictorial Data", IEEE Trans Comm, vol COM-22, pp. 614-624, May, 74.
2. Ahmed, N. et al, "Discrete Cosine Transform", IEEE Trans Computers, vol C-23, pp. 90-93, Jan, 74.
3. Murray, G. G., "Digital TV Microprocessor System", Proceedings of SPIE Conference, vol 119, Aug, 77.

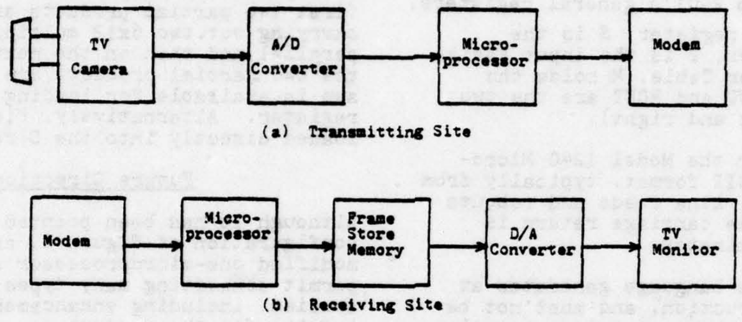


Figure 1 System Block Diagram

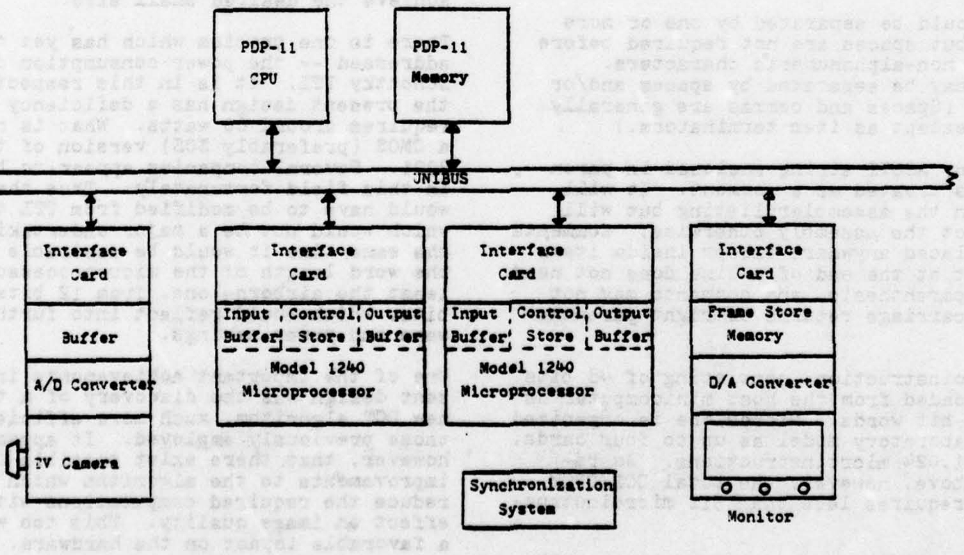


Figure 2 Laboratory System Block Diagram

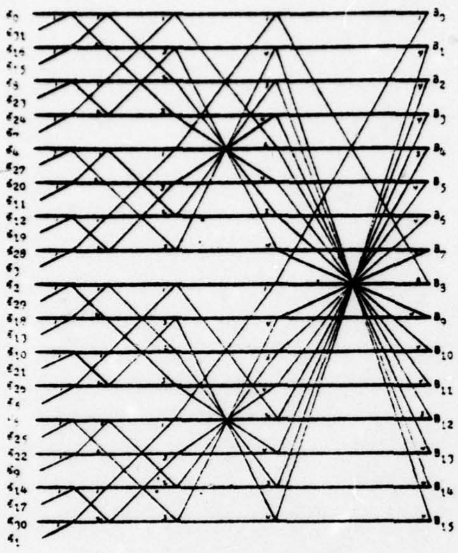


Figure 3 DCT Algorithm

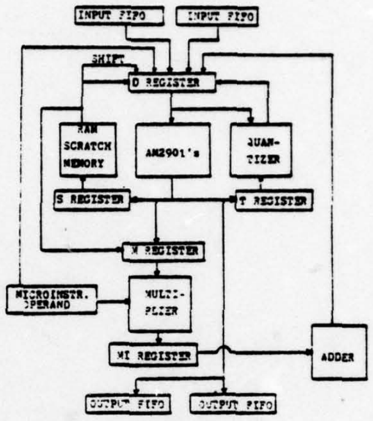


Figure 4 Model 1240 Microprocessor

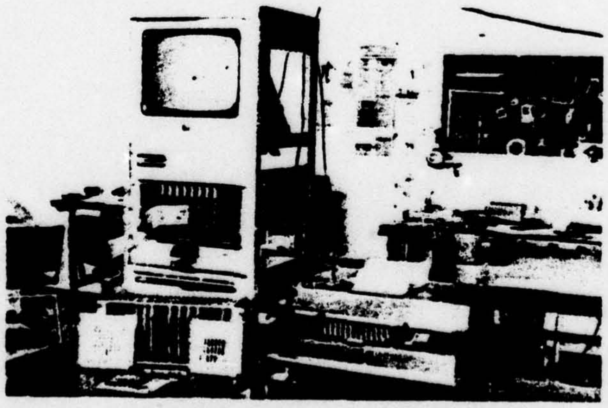


Figure 5 Laboratory System Photograph

APPENDIX E
APPLICATIONS
# of the
DIGITAL VIDEO PROCESSOR

T-116-1073

November, 1976

TABLE OF CONTENTS

LIST OF TABLES

SECTION I

INTRODUCTION

The digital video processor, Model D/W 1240, to be delivered to the AFAL early in 1977 is a powerful microprogrammed system which can be put to many uses. It will be accompanied by a FORTRAN cross-assembler to permit writing additional micro-instruction programs for future applications. The microstore, proposed to include 512 microinstructions in RAM, can readily be expanded to 4096 by plug-in cards. Also the microprocessor which is the heart of the digital video processor has been designed so that the PDP-11 can readily control it, either loading or reading out certain of the microprocessor internal registers.

The principal application intended is image processing, in particular data compression by means of the Discrete Cosine Transform/ Differential Pulse Code Modulation which experts in the field believe is near optimum in TV work. The microprocessor is a powerful design featuring the Advanced Micro Devices Am2901 4-bit Schottky TTL chip, which is fast becoming an industry standard. However, because it is microprogrammed, this device can be employed for all the familiar transforms used in image processing, such as the Hadamard-Walsh or straight DPCM.

But there is no restriction implied in the utilization of the Model D/W 1240. It represents a high-speed attachment to a minicomputer and hence can be programmed to process sonar or radar or FLIR data. Its main advantage is high-speed as it outruns the typical minicomputer by a factor of 20 or so in speed. Other possible applications include image enhancement, communications, encoding/decoding, pattern recognition, high-speed sorting, etc. Thus although the system has great value simply as a prototype for future Remotely Piloted Vehicle imagery applications, it also has other equally interesting applications.

SECTION II

IMAGERY APPLICATIONS

In the evolution of the digital video processor, two earlier configurations were judged to be inferior to that of Fig. E-1 -- Configuration C, which features both the Frame Store Memory and the TV Camera being accessible from the UNIBUS. The controller would be a PDP-11 minicomputer.

2.1 Configuration C

The system depicted in Fig. E-1 is capable of processing a vertical stripe 32 pixels wide in a single field time. The 32-pixel line is processed by the cosine transform (DCT), and then the coefficients from line to line are differentially encoded by means of the DPCM. This can result in data rates from the Remotely Piloted Vehicle in the range of 1600 down to 200 Kbps, which are well suited to spread spectrum communications with high processing gain to resist jamming.

Since an actual system would require two microprocessors -- one in the RPV and one on the ground -- Configuration C must make double use of the single microprocessor. This would be accomplished as follows. An entire field would be captured from the TV Camera and moved into the Frame Store Memory (FSM) along the UNIBUS. Then the microprocessor (assuming 1024 words of microstore) would hold both the forward DCT/DPCM and the inverse $DCT^{-1}/DPCM^{-1}$ transforms in microstore. It would then perform both the forward and the inverse transform on each line of 32 pixels transferred in and out from the FSM. When the entire field of 256x256 pixels was transformed in both directions, the original image on the Monitor would be replaced by the transformed and hence degraded image.

Note, however, that there is a 2-to-1 slow-down when a single microprocessor is employed. Since it takes two line times (2x 65 usec) to process a 32-pixel group in both the forward and inverse direction, the slow-down in field processing is from 60/8 to 60/16 per second. But sometimes this may not be acceptable.

2.2 Configuration D

In order to overcome the speed limitation of Configuration C, it is possible to add a second microprocessor, as shown in Fig. E-2. This configuration is a more accurate representation of an actual RPV application in which the two units on the left in Fig. E-2 represent airborne equipment and the two on the right, ground equipment. It is planned to design the Bus Interface Card of the microprocessor so that it can act both as Slave and Master. The first microprocessor (the one on the left) will fetch 32 pixels, packed into 16 words, either directly from the TV Camera buffer or from the FSM -- depending on the mode of operation selected by the Operator. After the first microprocessor performs the DCT and DPCM on this data, it is stored as 32 coefficients packed two per word, in its Output Buffer.

The second microprocessor will next act as Master and treating the first microprocessor as Slave will fetch the 32 coefficients. The inverse DCT and inverse DPCM operations will then be carried out. The reconstructed 32 pixels will be available in the Output Buffer. Again acting as Master the second microprocessor will store these pixels in the FSM for display.

Whereas in Configuration C it is necessary to first capture an entire field in a FSM so as to permit the single microprocessor to compress and expand the data in a slowed-down mode, with Configuration D this is not necessary. Data can be accessed directly from the TV Camera by the first microprocessor. This data, after processing, is passed to the second microprocessor. Finally it is moved to the FSM. This is equivalent to the moving of 16 16-bit words three times, or 48 word transfers during a line time of 65 usec. Even at 500 nsec per transfer, this would take only 24 usec. It should be noted that the microprocessor is computing as the data is being moved so that it has a full 65 usec either to compress or to expand the data.

The Controller always gives the first microprocessor a 65 usec headstart over the second one. Thus there is compressed data waiting for the second microprocessor when it is started.

2.3 Optional Modes

Certain optional imagery processing modes are possible. Included are the following, among others:

1. Hadamard-Walsh Transform combined with DPCM -- The Hadamard-Walsh Transform is actually simpler to implement than the Fast Fourier Transform or the DCT since no multiplications are involved.

2. Haar Transform -- This transform should also be capable of being performed since it has a rather simple algorithm.

3. Straight DPCM/DPCM -- Either single dimension or 2-dimensional DPCM can be employed.

4. Fast Fourier Transform -- This transform can also be used. Data/Ware has an excellent algorithm for applying the FFT to all-real input data.

2.4 Three-Dimensional Processing

Three-dimensional processing is extremely important since it offers the advantage of more powerful compression. One such scheme would require a 2-D DCT on each field followed by DPCM from field to field. In order to carry out such processing, an additional memory is needed in order to store the "coefficients from the preceding frame". Then the coefficients obtained by the DCT/DCT on the present frame can be differenced with those from the preceding frame.

Ideally there would be three FSM's in the system. The first would capture an entire field. The second would store the coefficients from the preceding frame. The third would refresh the Monitor. But for experimentation purposes three FSM's would not be required. With some ingenuity it should be possible to make do with core memory or with a small portion of the FSM. If the latter is used, then it would reduce the amount of FSM available for refreshing the Monitor. Hence only 1/2 of a normal picture from the TV might be seen. This might be sufficient for purposes of experimentation.

Three-dimensional processing requires "corner turning" in order to carry out the vertical DCT. After the horizontal DCT is performed on 32 pixels, the coefficients must be stored in a memory. The microprocessor is then able to address this memory and read out the data in blocks of 32 vertical pixels so as to do the second DCT. Hence there is need for 32x256 pixels or 4096 words which pack 2 coefficients per word. In doing 2-D transforms, it seems preferable to store one coefficient per 16-bit memory location. Thus there is need for 8192 memory locations.

Another requirement is for 256x256 memory locations for the coefficients of the <u>preceding</u> field. The compromise suggested would use half of the FSM for coefficient storage from the preceding frame. In addition 4096 words of core memory could be used for corner turning half of a stripe. A stripe would first be processed as a horizontal DCT and its coefficients would be stored in core memory. Next the vertical DCT would be carried out. Following that, the DPCM would be done with coefficients from the preceding frame stored in one half of the FSM. Then the inverse transforms are performed.

The final pixels to be displayed require that the inverse DPCM be done, which involves the present DCT/DCT coefficients and those of the preceding frame. Next the vertical inverse DCT is done -- still using the 4096 word working area in memory. Finally the horizontal inverse DCT is carried out and the pixels that result are sent to the FSM for normal display. Since half of the FSM (or even more) must be used for coefficient storage, the picture displayed must be a partial picture.

If in processing a stripe, only the top half of the TV picture is to be displayed and the lower half of FSM used for storing the preceding frame coefficients, then the core memory location for corner turning need only store 4096 coefficients rather than 8192. In essence the problem has been cut in half. Thus two microprocessors doing twice the work (horizontal followed by vertical DCT's) on half the data should be able to keep up.

2.5 Image Enhancement

Image enhancement can be achieved in several ways, and only one will be discussed here. An image, perhaps a reconnaissance photograph or a medical X-ray, is scanned. First a horizontal transform (DCT or FFT) is performed over 32 pixels, and this is followed by a vertical DCT or FFT. At this moment, 2-D coefficients are available. The PDP-11 controller can then manipulate these "frequency" coefficients to achieve some purpose. To reduce noise in a picture, the high-frequency coefficients can be set to zero; to enhance edges the high-frequency coefficients are increased in amplitude. If the picture is not pleasing to the eye because the distribution of gray scale values is too limited, they can be stretched to cover a wider range. After this manipulation, the inverse transforms are performed and the enhanced picture is recovered.

2.6 Playback of Reconnaissance Film

A technique which has been employed at NUC is to take a film taken from an aircraft and run it through a projector, which perhaps has been slowed down. The TV Camera and the projector are placed "face to face" through a tube. Thus the TV Camera now appears to be mounted in the aircraft. Real time processing can be carried out with very realistic results.

It is possible to experiment with different compression ratios, with different DPCM quantization, with modified algorithms, etc. As this is done, the reconstructed images on the Monitor can be examined. Clearly, such a set-up can be used to train operators and to obtain useful operational training in a laboratory environment.

2.7 FLIR and IR Imagery

Although the system has been described for TV processing, there is no reason why it cannot be adapted to FLIR and IR imagery applications. In these cases it is assumed that the emphasis is to be changed from compression to enhancement. This means that enhancement techniques described above may be applicable. Certainly a transform into the frequency domain of the DCT or FFT type followed by coefficient manipulation and the inverse transform offers the possibility of noise removal and image correction for sensor anomalies. Some airborne sensors operate in a mode which produces a single scanned line and the aircraft's motion is counted on to sweep out an area. The processing techniques discussed herein are well suited to such a situation.

2.8 Pattern Recognition

Because in piloted aircraft there are a large number of functions which must concern the pilot, it is difficult for him to fully utilize advanced sensors of the FLIR, IR, and low-level TV types. There is therefore a need for data processing which could assist the pilot in identifying man-made artifacts. For example, one discriminant frequently employed are straight lines -- such as might correspond to roads, buildings, etc.

Edge discrimination is consequently quite important and algorithms which assist in this process have been proposed by many investigators. Frequency domain computations are of considerable interest. It therefore should be possible to employ the digital video processor and the PDP-11 in investigations of this type. A suitable photograph can be placed in front of the TV Camera and in this way entered into the system for processing. Under software control in the PDP-11 the microprocessor can carry out a number of different algorithms for edge and straight-line enhancement in order to make them more visible.

SECTION III
FRAME STORE MEMORY UTILIZATION

Because in the system design the FSM was made accessible to the PDP-11 via the UNIBUS, flexibility of application is greatly increased. Furthermore, the system has been designed to permit the later addition of other FSM's.

3.1 Storing Images

In the ideal situation there would be at least two FSM's. The first would accept the raw imagery in digital form from the TV camera or other source. The second FSM would hold the processed pixels in order to update the Monitor. Note, however, in the text above there was a method described wherein the FSM could be employed also to store coefficients. This is possible because the FSM -- by being on the UNIBUS -- can readily be accessed by the microprocessor.

In order to permit the Monitor refresh function, the FSM must cycle in 400 nsec or less for a 16-bit word, equivalent to two pixels. In addition it must be possible to read from and to write into the FSM. Through special logic and overlapping of UNIBUS operations and memory operations, a throughput of 2.5 Megawords/sec for I/O is retained.

3.2 FSM as PDP-11 Memory

The FSM has been memory mapped so that it appears to the PDP-11 as a 4k portion of Main Memory occupying, say, the position from 24k to 28k. Although the FSM is actually 32k of 16-bit words, the PDP-11 Controller must set a field register within the FSM to identify the 4k currently being used. If two FSM's are being used they can be placed in different 4k ranges within Main Memory -- say 24k to 28k and also 20k to 24k. Alternatively, they can be given the same 4k range and the PDP-11 will identify which one can respond to a UNIBUS transaction by means of control bits in the Bus Interface Card between the FSM and the UNIBUS.

The above is necessary because the PDP-11 has only a 16-bit address word. But other devices on the bus are able to employ 18-bit addresses. Hence the FSM's are also assigned alternate 4k addresses in upper core memory which are unique. This makes it simpler when there are several FSM's present and the various microprocessors are addressing them simultaneously.

Since the PDP-11 can address the FSM, it is able to manipulate data therein directly. Hence it can investigate root mean square error in image reconstruction and carry out any similar desired computations. The advantage of this is that the desired data reduction on processed results can be carried out directly without having to resort to a large-scale computing center.

Because of the fact that the PDP-11 system lets the CPU and the Main Memory communicate over the UNIBUS as though they were any other Master-Slave pair, it is possible to utilize the FSM as normal PDP-11 memory. Thus it can store not only data but also programs. Thus there is an additional 32k of Main Memory available to the PDP-11 when imagery processing has been temporarily halted.

SECTION IV

OTHER ADVANCED APPLICATIONS

If the ability to microprogram the D/W 1240 processor is utilized fully, it would be possible to extend the system applications to many other fields. Some typical examples are briefly covered:

4.1 Fast Fourier Transform

Already mentioned above, the FFT is such a basic tool in engineering that it seems likely that this capability of the system can be exploited. However, one limitation is that the system is best suited to short data blocks. Longer ones would require some manipulation.

4.2 Modems and Communications

Digital communications systems often employ a number of subcarrier frequencies which are harmonics of a single tone. For example, in the Navy's Link 11 a carrier is modulated by 16 tones. Information in such communications channels is contained either in the presence or absence of such tones or in their phases. Hence a FFT can recover this information. Data/Ware has carried out studies which show the feasibility of using the D/W 1240 processor in this manner. Such a processor can act either as the modulator or demodulator in digital communications.

4.3 Voice Processing

Another application for a FFT processor is in voice processing, perhaps in the communications area. A processor in this type of application must be high-speed and able to do the FFT in order to compress digitized voice so that it can be sent at a relatively low data rate.

4.4 Encoding/Decoding

In order to perform error encoding/decoding a fast processor is required. Data/Ware has designed such units both for Reed-Muller and for BCH codes. Based upon this experience, it is known that the D/W Model 1240 can be so utilized.

4.5 Floating Point Processor

As an attachment to the PDP-11, the microprocessor can be used to perform special computations -- perhaps on matrices or in a floating point mode. There is one limitation, however, and that is the12-bit word length. For certain applications this might be acceptable in order to achieve high-speed processing.

4.6 Sorting

In certain military applications, it is necessary to sort data very quickly. Because of its high internal speed, the microprocessor should be able to sort data rather quickly. Data to be sorted would be sent by the PDP-11.

SECTION V
SUMMARY

The intended application of the Digital Video Processor System lies in imagery. Although designed and optimum in many respects for the DCT/DPCM data compression scheme on digitized TV data from a RPV, the system is capable of other image processing tasks, involving for example the Hadamard, Haar, straight DPCM, etc. Three-dimensional processing and image enhancement are also readily performed. Playback of 16mm film, FLIR and IR processing, and pattern recognition are other areas which can be investigated with this system. In a section on the Frame Store Memory it was pointed out that the FSM can also serve as PDP-11 Main Memory.

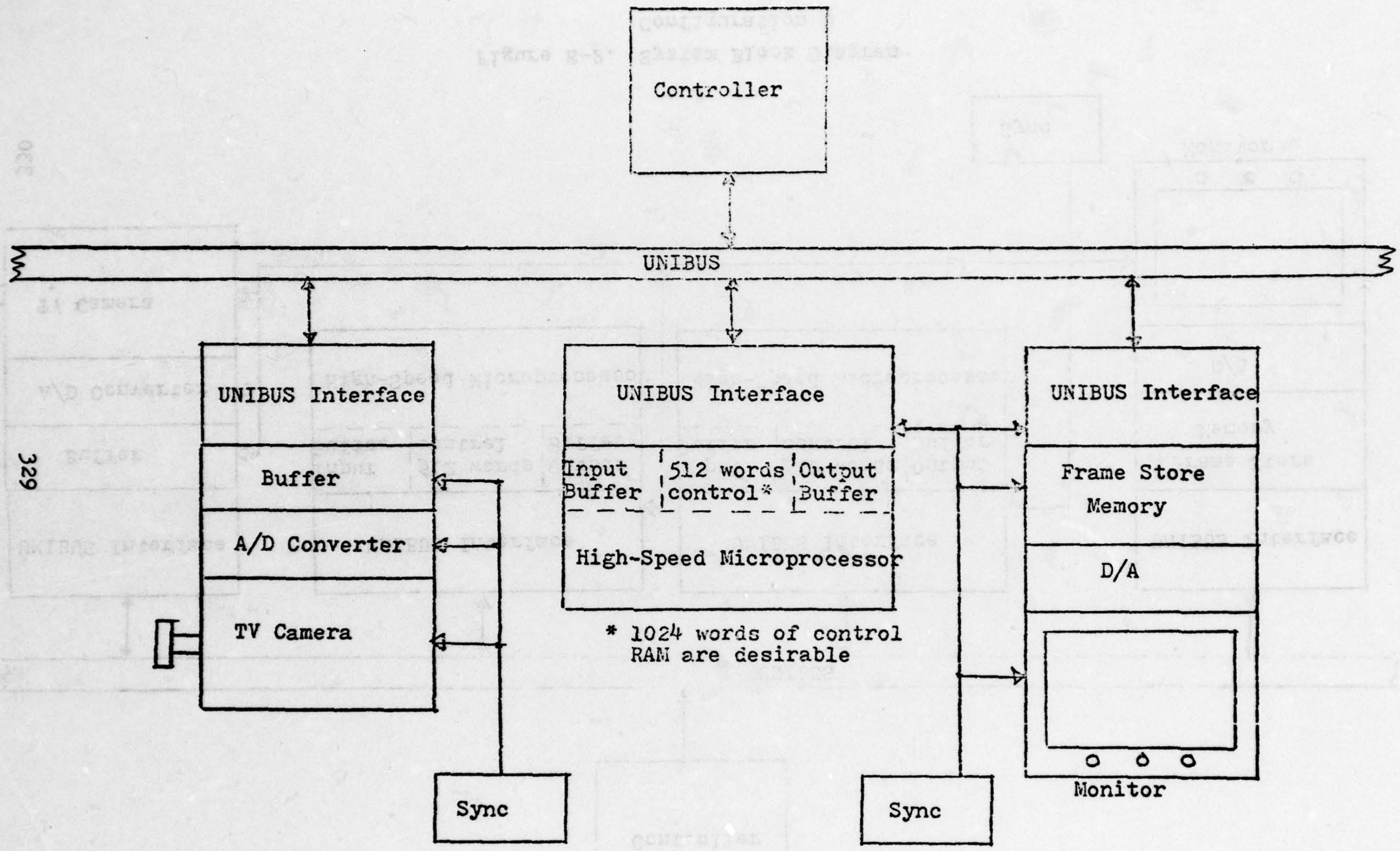


Figure E-1. System Block Diagram
Configuration C

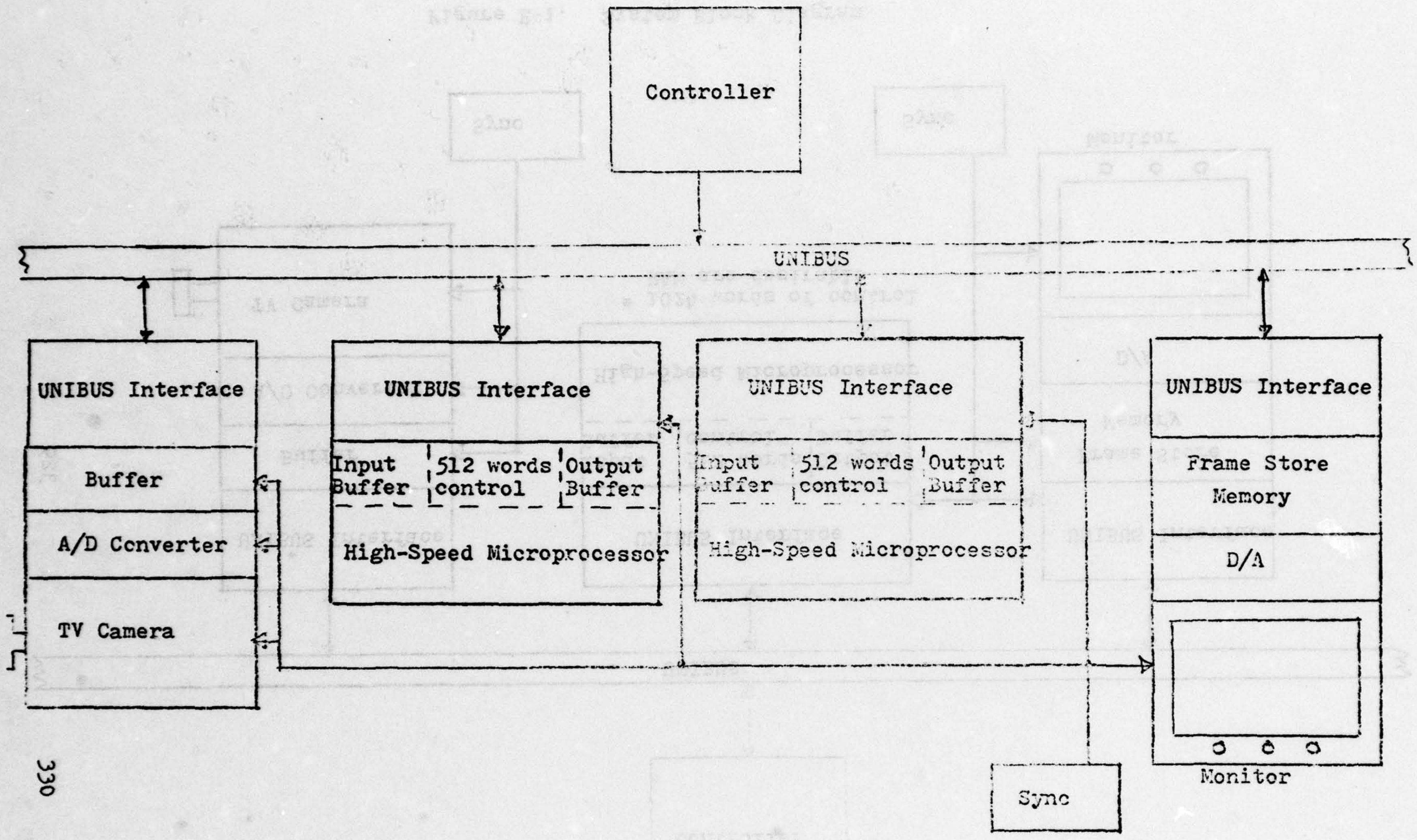


Figure E-2. System Block Diagram
Configuration D

APPENDIX F

MICROPROCESSOR SIZING

FOR

TV REDUNDANCY REDUCTION

T-46-940

April, 1976

TABLE OF CONTENTS

LIST OF TABLES

SECTION I

INTRODUCTION

In a report entitled, "TV Redundancy Reduction System Using A Bipolar Microprocessor", T-36-928, Data/Ware Development, Inc. described an implementation of the NUC Mini-RPV System using the Am 2901 bipolar microprocessor. An important consideration in this approach was the availability of a new, highly efficient algorithm developed by Data/Ware. Both the algorithm and the logic organization were outlined in the referenced report.

However, in order to assess the feasibility of the microprocessor-based system, it is important to estimate the size, weight, and power. The goals for these are a size of 4"x 8"x½", weight of 1 lb., and power consumption of 10 watts. The weight is not expected to be a problem, and so the issue resolves swiftly to one of size and power. Later it will be shown that size is not a problem either. Thus the final issue is one of power consumption.

Data/Ware is able to present a fairly accurate parts count based upon a Schottky TTL design. The reason for designing with this family was two-fold. First, the Am2901 4-bit bipolar slice is itself Schottky TTL. Second, this family is very high performance. In carrying out the design study referenced above, Data/Ware was not able to ascertain which parts of the design could be met by low-power Schottky TTL. Since this family requires only 1/10 the power of regular Schottky, it is extremely advantageous to make use of it. On the other hand certain parts of the design -- in the "critical" path -- must be left in Schottky TTL rather than in the low power version. Another trade-off is the use of the new I^2L family which is very low power yet high performance. This was not investigated either.

SECTION II

SELECTION OF WORD LENGTH

Because the word length of the basic Am2901 microprocessor is 4 bits, it is natural to narrow the choice of word length down to 8 bits versus 12. Harper Whitehouse has suggested that 9 bits should also be investigated since it can be achieved by two Am2901's plus additional logic for the sign. It turns out that the choice of 9 bits could very well be adequate. In particular, it is much superior to 8 bits. On the other hand, in the time available it was not clear that there would be a significant gain in parts reduction -- the reason being that so many IC's are available as 4-bit pieces and not as 5-bits. 8 bits appears to be marginal. This was the reason why Data/Ware has recommended 12 bits.(See Reference.)

It would require simulation in order to determine what the actual performance difference is between an 8-bit and a 12-bit imagery processing system. It is known that root mean square is not an accurate indication of the overall "acceptability" of a reconstructed image. On the other hand, it is necessary to resort to such a measure in order to permit an analytic approach

In the Data/Ware algorithm, 5 iterations of the FFT are required. On each, a doubling in size is possible. If the input data is 6 bits, then the output can grow to 11 bits. Hence with a 12-bit system, no scaling is required. But with a 9-bit system or an 8-bit system, 2 and 3 scalings, respectively, are required.

The effects of roundoff errors on the Discrete Cosine Transform due to scaling were calculated under the assumptions that the roundoff errors are independent and uniformly distributed and that the Central Limit Theorem can be applied. The analysis is based upon performing the DCT with roundoff errors, followed by performing the inverse DCT to great precision (as in a ground station) so that no additional errors are introduced. Thus the effect studied is the difference between the original image and the reconstructed image. The final rotation was not included.

Under the assumptions stated, it can be shown that the error in any one reconstructed input pixel is approximately Gaussian with zero mean and a variance that depends upon the kind of roundoff errors that occur. When no scaling is needed, the only roundoff error is due to multiplication by sines and cosines. Such multiplications can be rounded or truncated. When it is necessary to scale intermediate results by dividing by 2, this is performed during the last one, two, or three iterations. The effect of doing this can be computed.

Based upon the analysis performed, it is possible to prepare a table showing how the variance of the error depends upon the number of scalings and whether multiplication is truncated or rounded. Table F-1 summarizes the results.

TABLE F-1

ERROR ANALYSIS

Number of Scalings	**Variance of Error**	
	Rounded Multiplic.	**Truncated Mult.**
0	.0319	.1276
1	.0622	.1579
2	.2389	.4108
3	1.0456	1.6393

The significance of the Table is as follows. With the inputs digitized to 6 bits they range in value from 0 to 63. Adjusting the DC bias, they range from -31 to +32. Then, with a 9-bit processor and with rounded multiplications the variance in reconstructing the original image is .2389. It is quite possible that this is acceptable, but as noted above simulation would be necessary to confirm this. If only 8 bits are used, then 3 scalings are needed. This increases the variance to 1.0456, which is 4 times the previous value. Taking square roots, the standard deviation is doubled and remains near unity. Again, this might be acceptable.

SECTION III
SIZE AND POWER

Provided that the implementation selected is based upon "off-the-shelf" members of a digital logic family, the size problem can be solved by packaging uncased IC chips in hybrid modules. In a previous program for NELC, Data/Ware designed a BCH encoder/decoder of comparable complexity to the present redundancy reduction system which was packaged in 6 modules, each approximately 1.4"x1.8". The original design was the responsibility of Data/Ware, and NELC engineers carried out the partitioning. The NELC Microelectronics Laboratory developed the hybrids. Data/Ware participated in this program and was responsible for a checkout scheme for the hybrids using an Intel 4004 test system. Mr. Dean McKee of the Microelectronics Laboratory in a recent discussion described techniques for packaging up to 35 IC chips in very small hybrid packages, which are capable of dissipating 5 watts. Mr. McKee has expressed interest in the NUC program and has recommended a tour of his facility in order to demonstrate the capabilities of the Microelectronics Laboratory in hybridization, which is one of their functional areas in support of Navy programs. As a first estimate at the number of hybrid packages required, Data/Ware would recommend the use of 8 hybrids and would leave the four 40-pin IC devices as they are. Thus in all there would be 12 components to be mounted to the PC board, 8 of which would be hybrid packages of some 1.4"x1.8" and 4 of which would be standard 40-pin packages.

3.1 IC Families

The matter of power consumption is best addressed by considering some of the options available with respect to the IC family used.

1. TTL

This is a family with which Data/Ware, as is the case with most organizations, has had extensive experience. The Data/Ware Model 1640 FFT Unit is a high-performance Schottky TTL logic unit which can be clocked at 6 to 8 MHz. In the present redundancy reduction

design, Data/Ware again assumed the use of this family. This is natural since the A_m2901 4-bit microprocessor slice is itself Schottky TTL. Unfortunately power consumption is rather high. An informative comparison is low-power Schottky TTL vs. Schottky TTL. The former has a gate delay of 9.5 nsec and requires a power consumption of 2mW while the latter has a gate delay of 3 nsec and a power consumption 10 times as great. Since it is possible to mix the TTL families, an important consideration is to what extent Schottky TTL can be replaced by low-power Schottky TTL in the Data/Ware TV redundnacy reduction system. This has not been studied, but it should be stated that care must be exercised to assure that performance is not lost.

2. I^2L

"Heralded in some quarters as the answer to everything", I^2L or Integrated Injection Logic is being vigorously pursued at Texas Instruments, Fairchild, and Signetics. This family consumes little power and can operate over the entire military temperature range. TI has already introduced the I^2L 4-bit microprocessor slice, the SBP0400, with stated propagation times in the range from 110 to 530 nsec at 128 mW power. This represented their first standard product, and they are at present working on an improved performance model. By late 1976 TI is expected to have its advanced I^2L product line. Fairchild is also concentrating on high-performance I^2L. Thomas A. Longo, vice president, has stated that, "I^2L is the only really solid bipolar LSI technology. But being a bipolar technology, it should be used only where bipolar performance is required. That means memories operating under 100 nanoseconds and LSI logic with propagation delays of 10 nsec or less."

3. MOS

A review of the status of MOS technology was presented in the April 1, 1976 issue of Electronics entitled, "Advances in designs and new processes yield surprising performance". The new techniques include double polysilicon, V notch, double

diffusion, and charge coupling. With these new techniques, which are already being applied, it is anticipated that there will result, "two to three times greater speed, five and ten-fold increases in density. These designs threaten to steam-roll over bipolar large-scale integrated-circuit designs, to make MOS the dominant digital technology." Table F-2 from this article **surveys the various techniques available.**

TABLE F-2

LSI TECHNOLOGIES

Technology	Propagation delay (ns)	Power-delay product (pJ)	Density (Devices/mm^2)	Density (Gates/mm^2)	Chip size (mm^2)
High-threshold p-channel metal gate	80	450	150	50	7 x 7
p-channel silicon-gate	30	145	270	90	6.5 x 6.5
n-channel silicon gate	15	45	285	95	6 x 6
n-channel silicon gate depletion-load	12	38	320	107	6 x 6
n-channel double-polysilicon	10	35	525	175	6 x 6
Silicon-gate C-MOS	10	0.5	220	45	5.5 x 5.5
V-MOS D-MOS	5	20	600	225	–
SOS/C-MOS	2 – 5	0.1	650	275	5 x 5
I^2L (double level)	5 – 50	0.01 – 1	500	150	5.5

Looking at the more promising technologies, there is silicon-gate CMOS with the remarkable power-delay product of only 0.5. As an indication of improvements possible, RCA recently introduced a much higher performance "C^2L" microprocessor using self-aligned silicon gates. V-Mos uses a V-notch to increase device density. Electronic Arrays is working on V-notch ROMs, RAMs, and microprocessors. DMOS uses doubled-diffused doping to achieve gate delays as low as 1 to 5 nsec. Both V-MOS and D-MOS promise to achieve the speed of Schottky TTL. However, the most attractive technolgoies appear to be C-MOS on saphire and I^2L which combine high speed, low power, and high density.

3.2 Parts Count and Power

In the preceding section some approaches were presented as to reducing power consumption through a change in the logic family. It is possible to mix the families also. The exclusive use of Schottky TTL parts results in rather high power dissipation, as is shown in Table F-3. A 12-bit system (3 Am2901 4-bit slices) and an 8-bit system are presented. The 9-bit configuration, although deserving of study, would require very painstaking design since so many devices are 4-bit parallel or 2-bit parallel in organization.

Note that most of the watts -- 49.4 for the 12-bit system and 40.4 for the 8-bit system -- dissipated are in conventional devices, such as PROMS, latches, counters, and miscellaneous gates. If this part of the system could be designed in low-power Schottky TTL or in SOS/CMOS or in closed CMOS logic (C^2L) or in I^2L, a 10 times saving in power could be realized. Since these technologies either already have suitable devices available or in design for 1976 deliveries, their availability for a production program does not represent a problem.

SECTION IV

SUMMARY

Both an 8-bit processor configuration and a 12-bit configuration have been studied for the TV redundancy reduction system. The decision between the two would be based upon simulation results despite the fact that this report presented the variances to be expected in the reconstructed images. Regarding size, weight, and power of an all-digital implementation, it was shown that use of hybrid packaging of off-the-shelf IC's would readily meet size and weight constraints. However, the all-Schottky TTL design studied by Data/Ware presents a power dissipation problem which would require the use of some other family for standard logic parts. Candidate families were identified. Data/Ware would recommend an initial breadboard of Schottky TTL and a simultaneous investigation to choose the low-power replacement IC family.

TABLE F-3

DEVICES NEEDED AND POWER DISSIPATION

		12 Bit Configuration		8 Bit Configuration	
PINS		No. of Devices	Power (Watts)	No. of Devices	Power (Watts)
40	Am2901's	3	3.0	2	2.0
40	Multiplier	1	3.0	1	1.3
24	512X8 PROMS	13	7.8	13	7.8
16	64X4 FIFO	3	1.5	3	1.5
24	64X9 RAM	2	1.0	1	0.5
16	4-bit latches	34	15.3	23	10.3
16	Counters	8	4.0	8	4.0
16	Comparator	2	0.7	2	0.7
16	Shift Registers	2	1.5	2	1.5
14	2 to 1 MUX	7	2.2	5	1.6
14	4 to 1 MUX	7	1.3	5	1.1
16	Miscellaneous	23	4.6	23	4.6
	A/D	1	3.0	1	3.0
	S/H	1	0.5	1	0.5
		107	49.4	90	40.4

*U.S.Government Printing Office: 1979 — 657-002/23